BMC Remedy Action Request System 7.6.04

Integration Guide

January 2011

Contacting BMC Software

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

Address	BMC SOFTWARE INC 2101 CITYWEST BLVD HOUSTON TX 77042-2827 USA	**Telephone**	713 918 8800 or 800 841 2031	**Fax**	713 918 8000

Outside United States and Canada

Telephone	(01) 713 918 8800	**Fax**	(01) 713 918 8000

If you have comments or suggestions about this documentation, contact Information Design and Development by email at doc_feedback@bmc.com.

Restricted rights legend

Customer Support

You can obtain technical support by using the Support page on the BMC Software website or by contacting Customer Support by telephone or email. To expedite your inquiry, please see “Before Contacting BMC Software.”

Support website

You can obtain technical support from BMC Software 24 hours a day, 7 days a week at `http://www.bmc.com/support_home`. From this website, you can:

- Read overviews about support services and programs that BMC Software offers.
- Find the most current information about BMC Software products.
- Search a database for problems similar to yours and possible solutions.
- Order or download product documentation.
- Report a problem or ask a question.
- Subscribe to receive email notices when new product versions are released.
- Find worldwide BMC Software support center locations and contact information, including email addresses, fax numbers, and telephone numbers.

Support by telephone or email

In the United States and Canada, if you need technical support and do not have access to the Web, call 800 537 1813 or send an email message to `customer_support@bmc.com`. (In the Subject line, enter `SupID:<`*`yourSupportContractID`*`>`, such as `SupID:12345`.) Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC Software

Have the following information available so that Customer Support can begin working on your issue immediately:

- Product information
  - — Product name
  - — Product version (release number)
  - — License number and password (trial or permanent)
- Operating system and environment information
  - — Machine type
  - — Operating system type, version, and service pack
  - — System hardware configuration
  - — Serial numbers
  - — Related software (database, application, and communication) including type, version, and service pack or maintenance level
- Sequence of events leading to the problem
- Commands and options that you used
- Messages received (and the time and date that you received them)
  - — Product error messages
  - — Messages from the operating system, such as `file system full`
  - — Messages from related software

License key and password information

If you have a question about your license key or password, contact Customer Support through one of the following methods:

- E-mail `customer_support@bmc.com`. (In the Subject line, enter `SupID:<`*`yourSupportContractID`*`>`, such as `SupID:12345`.)
- In the United States and Canada, call 800 537 1813. Outside the United States and Canada, contact your local support center for assistance.
- Submit a new issue at `http://www.bmc.com/support_home`.

Contents

Preface 13

Audience 13
AR System documents 13

Chapter 1 What does "integration" mean? 17

Benefits. 18
Areas for integration 18
Real-time versus asynchronous integration. 18
Integrating with AR System 19

Chapter 2 Architectural overview of AR System 21

Terminology 22
    Form. 22
    Request 22
    Workflow 22
    Application 23
Multitier architecture. 24
AR System clients. 26
    Web client 26
    BMC Remedy User 26
    BMC Remedy Developer Studio 26
    BMC Remedy Data Import. 27
    BMC Remedy Alert 27
BMC Remedy Mid Tier 28
AR System server 28
Database servers. 29
Communications between clients and the AR System server 30
Communications between AR System servers and database servers . . . 30
Many-to-many connections 30
AR System components 32
Security and access control. 33

Chapter 3 Integration considerations 35

Where to integrate 36
    AR System client 36
    AR System server. 36

Database integration 36
Multiplatform issues 37
Choosing an implementation method 38
Integration technologies 38

Chapter 4 AR System C API 41

Overview 42
Understanding the AR System API 43
Program structure. 44
Multithreaded API clients 45
Using the API for integration 46
Example: Network management platform integration accessories 47
Issues and considerations 48

Chapter 5 Java API 49

Overview 50
Installed files 50
Contents of the AR System Java API installation 50
Run-time configuration 52
Java driver 52
Programming model 52
ARServerUser 53
Server objects 53
Cloning objects 53
Exception handling 53
Programming with the Java API 54
Sample 55

Chapter 6 Web services 61

Overview of web services in AR System. 62
Web service standards. 62
Predefined AR System web services 64
Forms and field mappings for web services 64
Basic and custom web services 65
Creating web service clients 66
Setting up the environment for web services 66
Verifying the AR System server configuration for web services 67
Configuring for a proxy server 67
Accessing WSDL or web services over https 68
AR System web services architecture 69
Information flow for web services published in AR System 69
Information flow for consuming a web service in AR System 71
Publishing a web service 72
Creating a web service. 72
Viewing a list of web services 80
Registering a web service 80
Web Services Registry prerequisites and configuration. 80
Managing web services registry entries 82

Synchronizing the Registry and Category forms with the registry 87
Exporting and importing data in the Registry and Category forms 88
Consuming a web service 89
Creating a Set Fields web service filter action 89
Querying the registry 91
WSDL limitations for consumption 94
Managing web service performance issues 94
SOAP headers and authentication 95
Authentication information for a published web service 96
Authentication information for consuming a web service 97
Supported schema constructs and AR System web service limitations 101
Supported XML schema constructs 101
The choice element 103
XML schema constructs not supported in AR System 103

Chapter 7 Plug-ins 105

About AR System plug-ins 106
Installing plug-in components 108
Installed files 109
Creating C plug-ins 110
C plug-in conventions 111
Creating Java plug-ins 112
Classes, instances, and shared data 112
Writing a Java plug-in 113
Configuring the Java plug-in server 114
Multithreading in the Java plug-in server 114
Dynamic plug-in loading 115
Plug-in naming conventions 117
Restarting the plug-in server using the Set Server Info command 118
Configuring the AR System server 118
Defining plug-in aliases 119
Running the plug-in server 120
Logging plug-in information 120
Common plug-in C functions and Java methods 123
Common C plug-in functions 123
Common Java plug-in methods 124
AREA plug-ins 124
AREA plug-in C API functions 126
AREA plug-in Java API methods 126
Sample AREA implementations 127
ARDBC plug-ins 127
ARDBC plug-in C API functions 128
Calling AR System API from an ARDBC plug-in 129
ARDBC plug-in Java API methods 130
Creating a vendor form for an ARDBC plug-in 130
AR filter API plug-ins 132
AR filter API plug-in C function 133
AR filter API plug-in Java methods 134

Chapter 8 LDAP plug-ins 135

Overview of LDAP and AR System 136
ARDBC LDAP plug-in 136
    Requirements 136
    Configuring the ARDBC LDAP plug-in 137
    Building AR System forms for directory services 140
    ARDBC LDAP run-time performance tips 144
AREA LDAP plug-in 145
    Configuring the AREA LDAP plug-in 145
    Configuring AREA LDAP group search 150
    Configuring AR System servers to use the AREA LDAP plug-in 152
    What's next? 152

Chapter 9 AR System external authentication 153

Overview of AREA authentication 154
    About the AREA LDAP plug-in 154
    Specifying AREA plug-in server settings 154
Configuring authentication processing 155
    Specifying when to use internal and external authentication 155
    Specifying authentication chaining mode 156
    Determining AREA behavior 157
    Configuring Atrium SSO integration 161
Setting up the AREA hub 164

Chapter 10 Data visualization fields 167

Overview 168
Services provided to the data visualization modules on BMC Remedy Mid Tier . . 168
Services provided on clients 170
Using Java classes 171
Working with native libraries 172
Storing shared library files on the mid tier 172
Creating data visualization fields 174
    Creating data visualization modules on the mid tier 175
    Registering data visualization modules 178
Configuring right-to-left format in a data visualization field (DVF) 182

Chapter 11 Vendor forms 183

About vendor forms 184
Creating vendor forms 184

Chapter 12 View forms 187

About view forms 188
    Database requirements for view forms 188
    AR System requirements for view forms 189
    Field properties for fields on view forms 189
    Database data types for view forms 189

Creating and modifying view forms 192
    Mapping an alternative AR System field type 194
    Modifying view forms 194
Setting up a remote database for view forms 194

Chapter 13 SQL database access 197

Accessing AR System data externally 198
Pushing data from AR System with SQL 198
Pulling data into AR System with SQL. 198
Issues and considerations 199

Chapter 14 ODBC database access 201

Overview 202
Creating multiple data sources 202
Compatibility with ODBC clients 205
Using Crystal Reports with AR System 205
    Using field labels or database names in Crystal Reports. 208
    Crystal Report report options considerations 209
    Selecting report fields in Crystal Reports 209
    Using Crystal Reports with join forms. 210
    Limitations when using Crystal Reports 210
Using Microsoft Access with AR System 211
Using Microsoft Excel with AR System 212
Issues and considerations 212

Chapter 15 Extending BMC Remedy Developer Studio 215

About extending BMC Remedy Developer Studio. 216
Prerequisites for creating plug-ins. 219
    Software requirements 219
    Project dependencies. 219
Extension points 219
BMC Remedy Developer Studio API 220
Installation directory 220

Chapter 16 BMC Atrium Integration Engine 221

The AIE integration with AR System 222

Chapter 17 BMC Atrium Orchestrator 225

Overview 226
The AR System Orchestrator Configuration form 226
    Modifying entries in the AR System Orchestrator Configuration form 228
AR System workflow for Atrium Orchestrator integration. 229
    Job status for asynchronous execution operations 232

Chapter 18 Exporting and importing data and definitions 233

Overview 234

Exporting objects and data to XML format. 234
  AR System objects in XML 234
  AR System data in XML 235
  Using XML with the AR System API 235
Using the import/export command-line utility. 235
  Guidelines for using the import/export utility 236
  DefinitionImport and DefinitionExport options 236
  Import/export examples. 239
Using the runmacro command-line utility 240
  runmacro example 242
Using the BMC Remedy Data Import utility 243
  Options for BMC Remedy Data Import command-line utility 245
  Importing in a multithreaded environment. 253
  Importing with a mapping file. 255
  Importing without a mapping file. 255
  Localization tips 255
  BMC Remedy Data Import utility examples 256
Using the BMC Remedy User CLI 257

Chapter 19 Running external processes (Run Process) 259

Overview 260
Client and server processes. 260
Using Run Process to start applications 260
  Example: open a reference document from an active link button 262
  Example: call a pager application from a filter 263
Using Run Process/$PROCESS$ to retrieve data from applications 264
Run a process on the web 266
  Limitations in using JavaScript 267
Issues and considerations 267

Chapter 20 OLE automation 269

OLE overview 270
AR System and OLE automation 271
Active links and OLE automation 272
  Using the GUID 273
  The OLE automation active link interface 273
  Reading the method tree. 275
Maintaining server context across multiple active link actions. 278
Working with ActiveX controls 278
AR System as an OLE automation server 278
DCOM support 279
The OLE automation active link action 279
  OLE automation example—Sample:SpellCheck 283
Issues and considerations 284

Chapter 21 Dynamic data exchange 285

Overview 286
  DDE parameters used by AR System 286

Methods of integration . . . 287
Configuring your system to use DDE with AR System . . . 287
Working with your dde.ini file . . . 287
DDE time-out settings . . . 290
Using active links with DDE . . . 290
Using the DDE active link action . . . 291
Using the DDE active link keyword . . . 293
Using BMC Remedy User reporting with DDE . . . 293
Configuring BMC Remedy User to pass report data . . . 294
Creating a report for DDE export . . . 294
Using a DDE execute from an external application to trigger AR System . . . 295
Using AR System with DDE, third-party applications, and macros . . . 296
DDE server name and BMC Remedy User path . . . 296
Supported DDE topic and function . . . 297
Example program and buffer . . . 297
Examples . . . 300
Integrating with Microsoft Excel . . . 300
Integrating with Microsoft Word . . . 303
Using DDE to pass data to Excel for graphing . . . 305
Issues and considerations . . . 306

Chapter 22 Simple network management protocol 307

Overview . . . 308
BMC Remedy SNMP Agent functions . . . 309
Monitoring AR System . . . 309
Sending traps . . . 310
SNMP configuration . . . 311
The arsnmpd configuration file . . . 312
System information . . . 313
Access control information . . . 313
Location of the armonitor configuration file . . . 316
The snmpd configuration file . . . 317
The armonitor configuration file . . . 318
Starting BMC Remedy SNMP Agent . . . 318
Stopping BMC Remedy SNMP Agent . . . 319
Troubleshooting . . . 319

Chapter 23 Making applications licensable for integration system vendors 323

Application licensing options . . . 324
Application licensing overview . . . 325
Making applications licensable . . . 326
Adding the application license to your server . . . 327
Assigning application licenses to users . . . 328

Appendix A Web service operation types 331

Create operation type . . . 332
Set operation type . . . 332
Get operation type . . . 333

Service operation type 334
XPATH function 335
Setting the start record and the maximum limit 337

Appendix B Mapping web service data 339

Mapping to simple and complex documents 340
Simple documents 340
Complex hierarchical documents 342
Mapping to complex documents 345
Using join forms in web services 350
XML editing 351
Simple XML editing 352
Object properties 352
Handling null, empty, and missing values 354
Advanced XML editing 358
Data types 360

Appendix C ARDBC LDAP example: Accessing inetorgperson data 363

Creating the inetorgperson vendor form 364
Attaching fields to represent inetorgperson data 365
Defining a filter to generate a DN 367
Summary of fields 368

Appendix D Web service examples 371

Example 1: Publishing a simple flat document 372
Example 2: Consuming a simple flat document 377
Example 3: Publishing a complex document 381
Example 4: Consuming a complex document 394

Appendix E Adding a certificate to a certificate database 401

About certificate databases 402
Creating a certificate database 402
Adding a certificate to a certificate database 403
Listing certificates in a certificate database 404

Appendix F AR System change ID utility 407

Introduction to archgid 408
Syntax of archgid 412
Using archgid in prompt-driven mode 415
Using archgid in bulk mode 417

Index 419

Preface

IMPORTANT

The compatibility information listed in the product documentation is subject to change. See the compatibility matrices at `http://www.bmc.com/support` for the latest, most complete information about what is officially supported. Carefully read the system requirements for your particular operating system, especially the necessary patch requirements.

Audience

This guide is written for developers and administrators responsible for creating, customizing, and maintaining integrations between BMC Remedy Action Request System (AR System) and external systems.

Before you read this guide, you should have a strong working knowledge of AR System, BMC Remedy Developer Studio, and BMC Remedy User. In addition, it is helpful to have a working knowledge of the external systems you are considering for integration with AR System.

AR System documents

The following table lists documentation available for AR System 7.6.04.

Unless otherwise noted, online documentation in Adobe Acrobat (PDF) format is available on AR System product installation DVDs, on the Customer Support website (`http://www.bmc.com/support`), or both.

You can access product help through each product's Help menu or by clicking Help links.

NOTE

The AR System product help has not been updated for version 7.6.04. The help topics still apply to version 7.6.03. For the most recent content, refer to the PDF documentation.

Title	Description	Audience
Concepts Guide[1]	Overview of AR System architecture and features; includes information about add-on products that extend AR System functionality and a comprehensive glossary for the entire AR System documentation set.	Everyone
Installation Guide	Instructions for installing AR System.	Administrators
Introduction to Application Development with BMC Remedy Developer Studio	Information about the development of AR System applications, including an introduction to using BMC Remedy Developer Studio.	Developers[2]
Form and Application Objects Guide	Information about AR System applications and their user interface components, including forms, fields, views, menus, and images.	Developers
Workflow Objects Guide	Information about the AR System workflow objects (active links, filters, and escalations) and how to use them to create processes that enforce business rules.	Developers
Configuration Guide	Information about configuring AR System servers and clients, localizing, importing and exporting data, and archiving data.	Administrators
BMC Remedy Mid Tier Guide	Information about configuring the mid tier, setting up applications for the mid tier, and using applications in browsers.	Administrators
Integration Guide	Instructions for integrating AR System with external systems by using web services, plug-ins, and other products, including LDAP, OLE, and ARDBC.	Administrators / Developers / Programmers[3]
Optimizing and Troubleshooting Guide	Information about monitoring and maintaining AR System and AR System applications to optimize performance and solve problems.	Administrators / Developers / Programmers
Database Reference	Database administration topics and rules related to how AR System interacts with specific databases; includes an overview of the data dictionary tables.	Administrators / Developers / Programmers
BMC Remedy Distributed Server Option Guide	Information about implementing a distributed AR System server environment with BMC Remedy Distributed Server Option (DSO).	Administrators
BMC Remedy Flashboards Guide	Instructions for creating, modifying, and administering flashboards to display and monitor AR System information.	Administrators / Developers
C API Reference	Information about AR System data structures, C API function calls, and OLE support.	Programmers
C API Quick Reference	Quick reference to C API function calls.	Programmers

Title	Description	Audience
Java API	Information about Oracle Java classes, methods, and variables that integrate with AR System. For the location of the JAR file containing this online documentation, see the information about the Java API in the *Integration Guide.*	Programmers
Java Plug-in API	Information about Java classes, methods, and variables used to write plug-ins for AR System. For the location of the JAR file containing this online documentation, see the information about plug-ins in the *Integration Guide.*	Programmers
BMC Remedy Email Engine Guide	Instructions for configuring and using BMC Remedy Email Engine.	Administrators
Error Messages Guide	Descriptions of AR System error messages.	Administrators / Developers / Programmers
Master Index	Combined index of all books.	Everyone
BMC Remedy Approval Server Guide	Instructions for using BMC Remedy Approval Server to automate approval and signature processes in your organization.	Administrators
Release Notes	Information about new features, compatibility, and international issues.	Everyone
Release Notes with Known Issues	Information about new features, compatibility, international issues, installation planning, and open issues.	Everyone
BMC Remedy User Help	Instructions for using BMC Remedy User.	Everyone
BMC Remedy Developer Studio Help	Instructions for using BMC Remedy Developer Studio to develop AR System forms, workflow objects, and applications.	Developers
BMC Remedy Data Import Help	Instructions for using BMC Remedy Data Import.	Administrators
BMC Remedy Alert Help	Instructions for using BMC Remedy Alert.	Everyone
BMC Remedy Mid Tier Configuration Tool Help	Instructions for configuring BMC Remedy Mid Tier.	Administrators
BMC Remedy Browser Help	Instructions for using AR System forms in browsers.	Everyone
BMC Remedy Migrator 7.6.04 BMC Remedy Migrator Guide	Outlines procedures for installing BMC Remedy Migrator, setting options, and performing migration tasks.	Administrators / Developers
BMC Remedy Migrator online help	Procedures for setting BMC Remedy Migrator options and performing migration tasks.	Administrators / Developers
BMC Remedy Encryption Security 7.6.04 BMC Remedy Encryption Security Guide	Provides an overview of the BMC Remedy Encryption Security products and explains how to install and configure them.	Administrators

[1] The full title of each guide includes *BMC Remedy Action Request System 7.6.04* (for example, *BMC Remedy Action Request System 7.6.04 Concepts Guide*), except

the *BMC Remedy Migrator Guide* and *BMC Remedy Encryption Security Guide*.
[2] Application developers who use BMC Remedy Developer Studio.
[3] C and Java programmers who write plug-ins and clients for AR System.

Chapter

1 What does "integration" mean?

In the context of software applications, integration means linking products together to provide increased functionality and utility. In other words, two products together do more (or do it faster) than the products by themselves.

AR System is a powerful foundation and development environment for applications that automate business processes. Its flexible multiplatform, multidatabase architecture and highly customizable user interface enable AR System to be adapted to the unique business processes of a particular company and to evolve as those processes change. However, AR System alone cannot perform all of the functions in an environment. Instead, AR System applications can be integrated with other applications and tools to form complete business solutions.

AR System is an open system, with many interfaces for linking to other products. This document provides an overview of AR System and these interfaces, and discusses methodologies for creating integrated environments.

The following topics are provided:

- Benefits (page 18)
- Areas for integration (page 18)
- Real-time versus asynchronous integration (page 18)
- Integrating with AR System (page 19)

Benefits

The primary intent of business software is to enable users to do their jobs more quickly with fewer resources. Using two products separately is usually less efficient than using them in an integrated fashion. For example, a user might have to enter the same information into two different applications, which often results in errors. Or the telephone number of an incoming call might be manually entered by a customer service representative rather than automatically captured. Application integration can provide improved efficiency and effectiveness.

Areas for integration

The two primary areas for integration between applications are:

- **Data sharing**—Passing data structures back and forth or jointly accessing a common database.
- **Process linking**—One application (App1) automatically launches another (App2) "in context" so that App2 "knows" everything entered into App1, and the user is immediately focused at the part of App2 that continues the process. Or App2 automatically does its job in the background based on directions from App1, and the user does not even know it is running.

  The overall environment behaves as if it were one large application, and yet the company can choose the discrete pieces that best meet the business requirements.

Real-time versus asynchronous integration

Products are sometimes integrated for "real-time" interaction. For example, in a help desk environment, a user calls a support person with a question. During the call, the support person enters information about the user and the question into the call tracking application. If the best way to answer the question is for the support person to walk the user through a process on the user's workstation, the support person could click a button on the call tracking application interface that runs a remote control application. The remote control application opens a window on the support person's workstation that is a copy of the user's screen, and the support person can take control of the keyboard and mouse functions of the user's system to step through a process. The user gets an answer and the support person never leaves his or her desk.

In contrast, some integration is done "asynchronously." This means that an application can be updating another application on an ongoing basis so that the second application is up-to-date the next time it is accessed. For example, suppose a Human Resources application contains the names and office numbers of all of the current employees of a company. Every night, the HR application writes a file that contains an alphabetical list of all of the employees to a defined place on a file server. Whenever the help desk starts the call tracking application, the application reads this file and dynamically builds menus of the employee names so that the support personnel can fill in their forms quickly. Conversely, whenever a change request to move an office is processed by the help desk, a notification is sent to the HR system that contains the affected employee name, the new office number, and an effective date.

Integrating with AR System

AR System is a platform on which you can build applications for automating a wide range of support and business processes. In many IT organizations, AR System-based applications are the central applications for tracking information. Therefore, the opportunities to integrate AR System with other applications are endless, ranging from simple access to diagnostic utilities to large-scale integration with manufacturing, customer interaction, and financial accounting systems.

NOTE

A hallmark of AR System is its rich and robust API. All prospective product partners are encouraged to integrate with AR System at the API level whenever possible. For more information, see the *C API Reference*.

Many customers purchase AR System as a development platform to create their own business applications and automate their business processes. BMC Remedy also develops and sells specific applications such as BMC Remedy Help Desk™, BMC Remedy Asset Management™, BMC Remedy Customer Support™, BMC Remedy Quality Management™, or BMC Remedy Change Management™. These BMC Remedy applications are built on top of AR System.

Integration at the API level is encouraged because your integration can be more easily adapted to BMC Remedy customers who utilize applications purchased from BMC Remedy and can be adapted to those BMC Remedy customers who build their own custom applications. BMC Remedy User is built on the AR System C API. BMC Remedy Mid Tier and BMC Remedy Developer Studio are built on the AR System Java API.

Chapter 2

Architectural overview of AR System

AR System has a multitier client/server architecture. AR System clients provide user interface facilities available from various platforms, including the Web. This section discusses the terminology and overall architecture of AR System.

The following topics are provided:

- Terminology (page 22)
- Multitier architecture (page 24)
- AR System clients (page 26)
- BMC Remedy Mid Tier (page 28)
- AR System server (page 28)
- Database servers (page 29)
- Communications between clients and the AR System server (page 30)
- Communications between AR System servers and database servers (page 30)
- Many-to-many connections (page 30)
- AR System components (page 32)
- Security and access control (page 33)

Terminology

The following terms describe the architecture of AR System.

Form

The primary user interface is a *form*, which is a screen made up of fields. Fields can be required, optional, or system-filled**.** Each field can have some type of data entered into it, including text, numbers, and dates and times. An example of a form is shown in Figure 2-1 on page 23.

A form corresponds to a table in a database, and the fields on a form correspond to columns in the tables. (Because forms were called "schemas" before AR System 4.0, many API calls still use that term.)

Request

When a user fills in the required fields on a form and tells the system to save the information in the database, a new *request* is created (or submitted). A request is one row of information in the database. It might be a trouble ticket or change request. (Because requests were called "entries" before AR System 4.0, many API calls still use that term.)

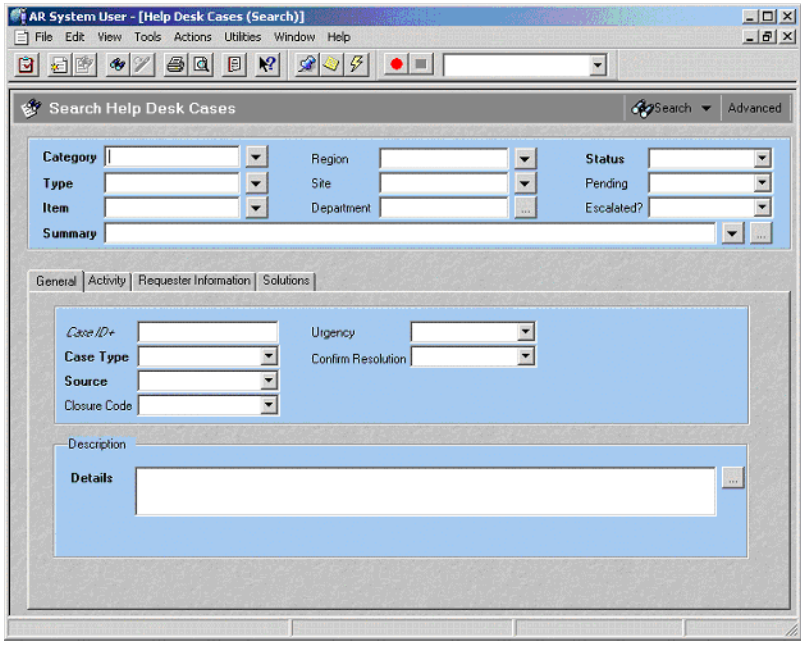
Workflow

When data is written into a database or into a form on the client, it sits there until someone or some application decides to change it. *Workflow* provides a means to take action based on this data, and the actions can result in changes to the data. Workflow is composed of descriptions (sometimes called "definitions" or "rules") that specify the actions to take and the conditions that can trigger them. Workflow can be triggered by the state of data (for example, the value in a field) or by the amount of time the data is stored. Actions can include running other applications, notifying people, changing data in the database or on-screen, and running reports.

Application

AR System is used to track information such as trouble tickets, change requests, asset records, purchase orders, stock trades, and service level agreements. A complete trouble ticket *application* might consist of a main form that contains the caller identification, problem description and work log information, and several secondary forms that are "linked" or "related" to the main form. Examples of secondary forms are an employee detail form that contains previously entered records about all of the employees at the company (for example, name, phone number, office location, PC type, and so on), and a solution form that contains summaries of previously entered problems and solutions. When a new trouble ticket is entered into the form, AR System workflow can look up information in the other forms and add it to the trouble ticket. This way, common information does not need to be reentered multiple times. BMC Remedy has developed a number of applications, including BMC Remedy Help Desk, BMC Remedy Asset Management, BMC Remedy Change Management, BMC Remedy Service Level Agreements, BMC Remedy Quality Management, BMC Remedy Customer Support, and BMC Remedy Citizen Response.

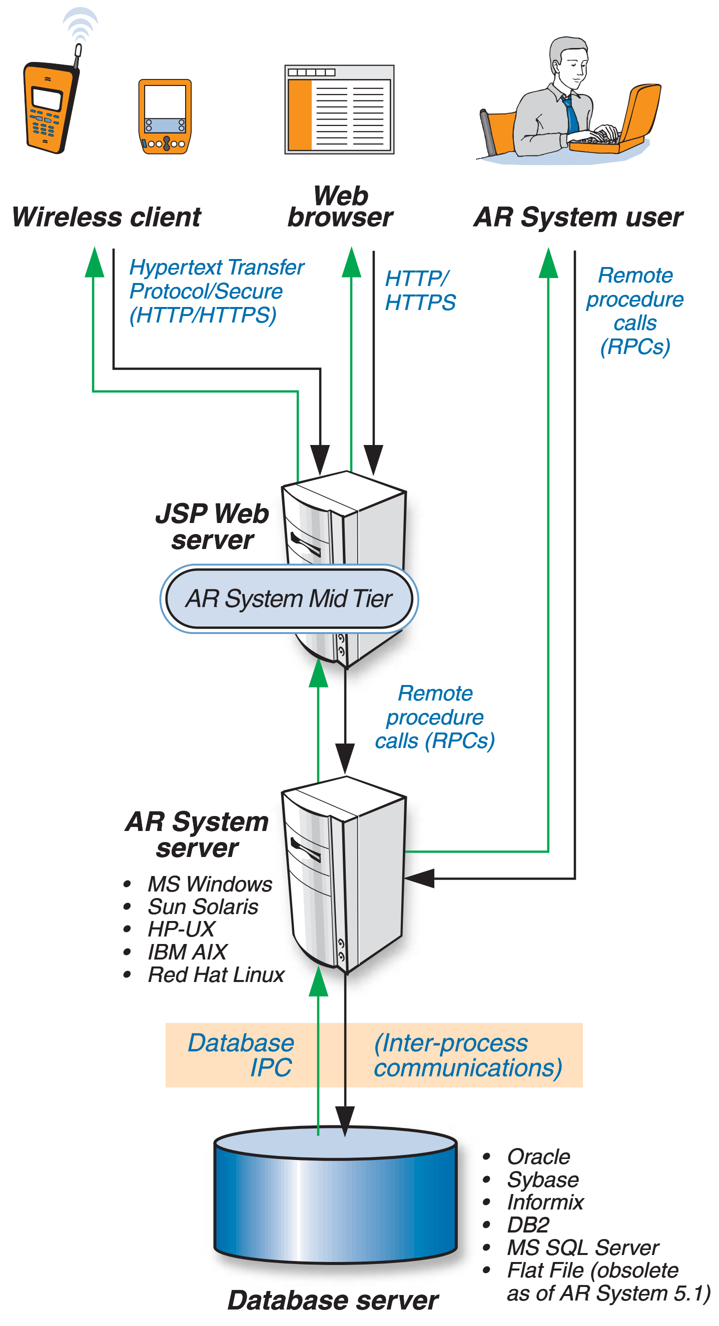
Figure 2-1: Sample BMC Remedy form



Multitier architecture

AR System is a multitier client / server architecture (see Figure 2-2 on page 25). AR System clients provide the user interface. The AR System mid tier makes the user interface available in browsers. The AR System server implements the workflow functions, access control, and flow of data into and out of the database. The database server acts as a data storage and retrieval engine. (For information about supported platforms, see the compatibility matrices at `http://www.bmc.com/support`.)

Figure 2-2: AR System multitier architecture



AR System clients

AR System clients are available for a number of operating system environments, as listed in Figure 2-2 on page 25. For each operating systems, the client is composed of a set of native applications (tools) that use the standard user interface conventions for that environment. Individual users can run these tools as necessary. Each client includes one or more of the following tools.

Web client

Through the BMC Remedy Mid Tier, users can access AR System in a browser. Using the web-based interface, users can submit and modify new requests, to search for information about requests, and to generate reports. Web pages are written in JSP™ and rendered in JavaScript™ and HTML.

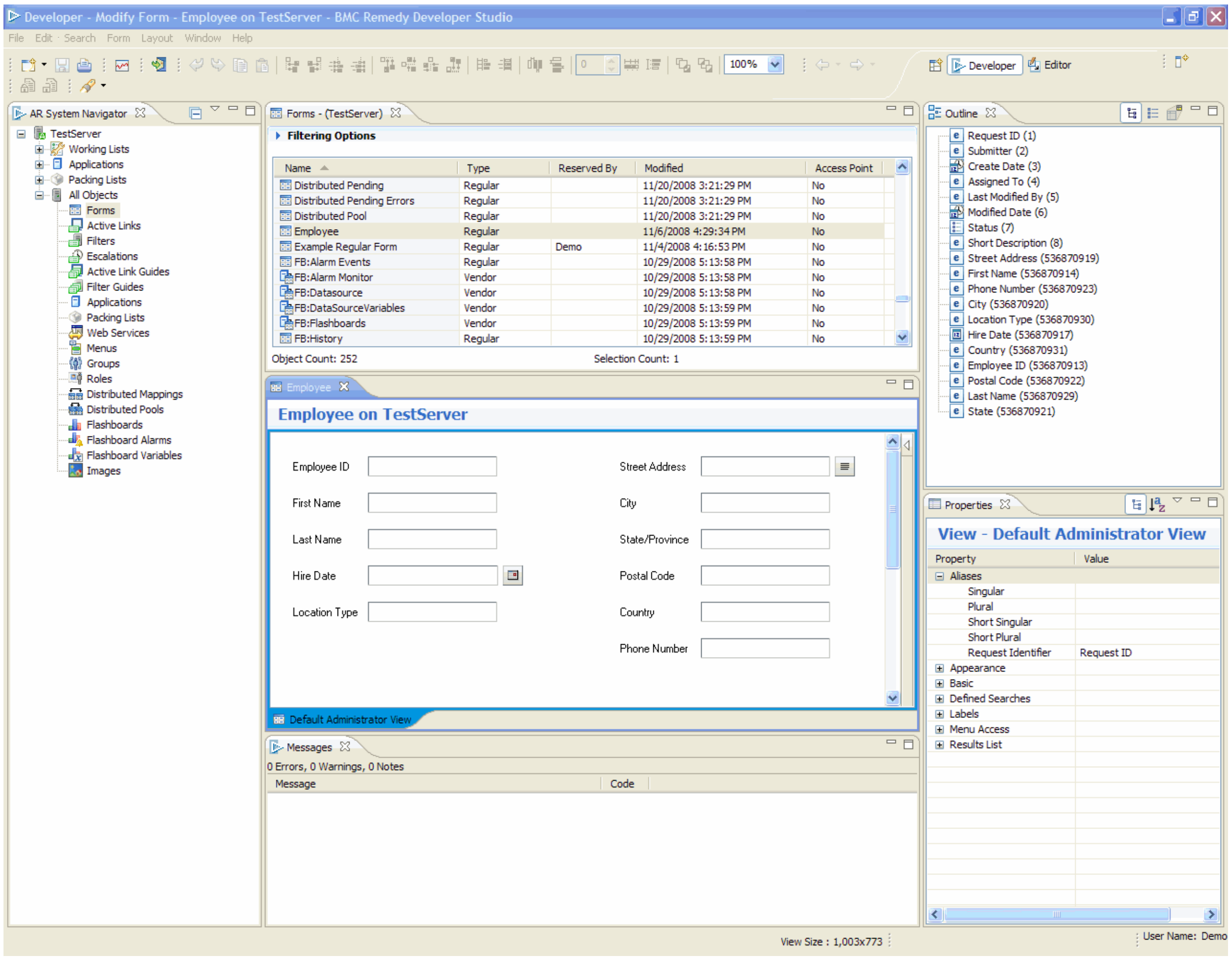
BMC Remedy User

A executable client for the Windows platform that provides an interface to AR System applications. It is also used to submit and modify new requests, to search for information about requests, and to generate reports. An example of the interface is shown in Figure 2-1 on page 23.

BMC Remedy Developer Studio

Used by AR System administrators to create and modify applications. All the components that make up an application, such as forms and workflow definitions, are created and modified using BMC Remedy Developer Studio. There is no programming or scripting language used. An example of the interface is shown in Figure 2-3 on page 27. BMC Remedy Developer Studio requires a Windows platform.

Figure 2-3: BMC Remedy Developer Studio interface



Name	Type	Reserved By	Modified	Access Point
Distributed Pending	Regular		11/20/2008 3:21:29 PM	No
Distributed Pending Errors	Regular		11/20/2008 3:21:29 PM	No
Distributed Pool	Regular		11/20/2008 3:21:29 PM	No
Employee	Regular		11/6/2008 4:29:34 PM	No
Example Regular Form	Regular	Demo	11/4/2008 4:16:53 PM	No
FB:Alarm Events	Regular		10/29/2008 5:13:58 PM	No
FB:Alarm Monitor	Vendor		10/29/2008 5:13:58 PM	No
FB:Datasource	Vendor		10/29/2008 5:13:58 PM	No
FB:DataSourceVariables	Vendor		10/29/2008 5:13:59 PM	No
FB:Flashboards	Vendor		10/29/2008 5:13:59 PM	No
FB:History	Regular		10/29/2008 5:13:59 PM	No

Object Count: 252 Selection Count: 1

0 Errors, 0 Warnings, 0 Notes

Message	Code

- Request ID (1)
- Submitter (2)
- Create Date (3)
- Assigned To (4)
- Last Modified By (5)
- Modified Date (6)
- Status (7)
- Short Description (8)
- Street Address (536870919)
- First Name (536870914)
- Phone Number (536870923)
- City (536870920)
- Location Type (536870930)
- Hire Date (536870917)
- Country (536870931)
- Employee ID (536870913)
- Postal Code (536870922)
- Last Name (536870929)
- State (536870921)

View - Default Administrator View

Property	Value
⊟ Aliases	
Singular	
Plural	
Short Singular	
Short Plural	
Request Identifier	Request ID
⊞ Appearance	
⊞ Basic	
⊞ Defined Searches	
⊞ Labels	
⊞ Menu Access	
⊞ Results List	

View Size : 1,003x773

User Name: Demo

BMC Remedy Data Import

Used to load external data into the AR System database. For example, employee information could be extracted from a Human Resources application and loaded into a Company People Info form as a batch process, eliminating the need to retype any data.

BMC Remedy Data Import provides a graphical interface that allows the building of the mapping between the columns in the external data set and the fields in the AR System form. These mapping definitions can be saved and reused. BMC Remedy Data Import is available for Windows.

BMC Remedy Alert

Alerts users to events. For example, it can display a message informing help desk personnel that a new problem was assigned to them. Clicking a notification opens a ticket in BMC Remedy User.

BMC Remedy Alert is a small application that is typically run as a background process on an AR System user's workstation. From BMC Remedy Alert, you can open up a list of alerts in BMC Remedy User or in a browser. BMC Remedy Alert shows only a summary of the number of alerts received. BMC Remedy Alert runs only on Windows platforms.

In addition to these applications, some additional processes act as clients in AR System. These include the BMC Remedy Migrator change automation product, the Network Management Platform Integration accessories, and the Systems Management Integration utilities. These are independent of the standard user desktop tools. Several of these products and the mechanisms they use for integration with AR System are described in Chapter 4, "AR System C API."

BMC Remedy Mid Tier

The mid tier translates client requests, interprets responses from the server, handles web service requests, and runs server-side processes that bring AR System functionality to web and wireless clients. For example, unlike BMC Remedy User, a browser is a generic client that has no inherent knowledge of any application that might run within it. By acting as an interpreter, the mid tier allows a browser to become a fully functional AR System client.

The BMC Remedy Mid Tier requires a supported Java Server Pages (JSP) engine. Tomcat is bundled with the mid tier, and is installed with the mid tier as part of the mid tier installation by default.

For the latest information about supported platforms and software, see the Remedy compatibility matrices at `http://www.bmc.com/support`.

AR System server

The AR System server is a set of processes that run on an application's server host. The AR System server is available for Windows and for a variety of UNIX® versions, as shown in Figure 2-2 on page 25. The server is implemented as several service processes that are normally started automatically when the server host is powered up.

The server processes have no direct user interface. They communicate with other processes, either AR System clients or external applications, through an application programming interface (API). An API is one possible way to integrate with AR System.

The AR System server processes the data that is enter by way of an AR System client. It writes the data into the database when a new record is submitted, and retrieves that data when a client makes a query. The server checks that a user has permission to do each requested transaction, enforcing any access control defined as part of an application. The server also continuously evaluates the data in the database and each transaction to determine whether any workflow should be triggered.

These are the key components of the AR System server:

- `arserverd` **process**—The primary AR System server process. It handles requests from the clients and interacts with the database.
- **plug-in server**—A companion process to the AR System server. It loads configured plug-in modules to interface with external data while processing vendor forms, validates users with external authentication sources, and extends filter workflow. When the AR System server needs to access a plug-in, it interfaces with the plug-in server to perform the operation.
- **Email process**—Java on UNIX or `aremaild` on Windows; the process that links `arserverd` with email systems. For example, users can submit service requests to an AR System server by sending an email message to an email account. In addition, workflow can cause email messages to be sent to particular email addresses as a notification option.

Database servers

AR System uses standard relational database engines for the actual storage and retrieval of data. Architecturally, the database server is an independent set of processes that are completely separate from the AR System server processes. Physically, the database server processes can be running on the same server host as the AR System server or on a different host. The database server can be any platform that the database engine supports.

AR System is not a database application in the typical sense. All of the workflow is managed by the AR System server, so proprietary database features such as triggers and stored procedures are not used. An application created on an AR System server running one type of database engine can easily be moved to a server running a different database engine through a simple export/import process.

The user or administrator of AR System clients does not need to know anything about SQL or the underlying database.

Communications between clients and the AR System server

All clients of the AR System server communicate with the server by using remote procedure calls (RPCs) on top of a TCP / IP transport stack. The type of RPC is the Oracle ONC™ RPC.

TCP / IP networks are the de facto standard for corporate and Internet communications. The RPC mechanism is used because it is a “lightweight” transport that uses minimal network bandwidth, yet provides robust communications services. It can function over slower dial-up network links and high-speed internets and intranets, and is supported over most of the wireless networking technologies.

The AR System web server communicates with the browsers using HTTP (Hypertext Transfer Protocol) or HTTPS (Secure HTTP).

Communications between AR System servers and database servers

From the perspective of the database server, the AR System server is a database “client.” BMC Remedy uses the database client libraries from the various databases. When an AR System server is installed, the installer specifies the type and location of the database server, and the proper database client is enabled. AR System servers communicate with the database servers through whatever inter-process communications (IPC) mechanism the database client uses. Examples include SQL*Net for Oracle and OpenClient for Sybase.

Some database engines are multithreaded. This means that the database can perform multiple transactions simultaneously. In AR System, the `arserverd` server process is also multithreaded. Each of these `arserverd` threads is connected to a different thread in the database engine**.** This provides tremendous data throughput and system scalability.

Many-to-many connections

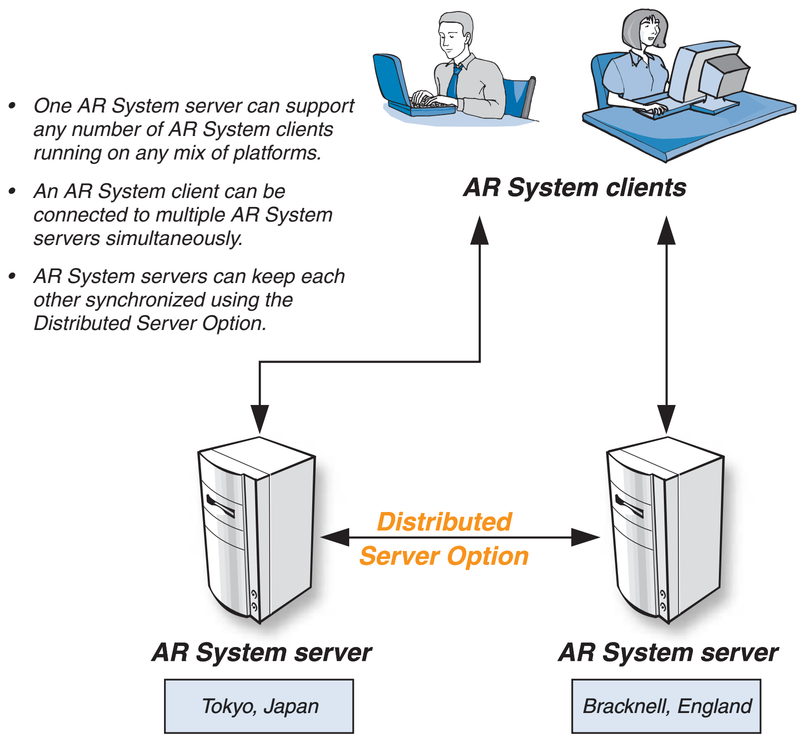
In an AR System environment, one AR System server can theoretically support any number of AR System client connections (limited by network bandwidth and server host and database performance). The clients can be on any mix of platforms.

Similarly, an AR System client can be connected to any number of servers at the same time. These servers can be any mix of server hosts and underlying database engines.

In an environment with multiple AR System servers, the Distributed Server Option (DSO) can be added to share common information among servers and keep that information consistent. DSO also enables records to be forwarded between servers if workflow determines that the record should be transferred. This permits building large-scale distributed support environments that behave as a single virtual system. Some of the many uses of DSO include:

- Transferring requests to the location at which they can be processed.
- Transferring requests between regions in a 24-hour, 7-days-per-week customer support environment.
- Creating a distributed knowledge base so that problem-solving information can be referenced at any location.
- Archiving old requests to a reporting server to maximize the performance of the production server.

Figure 2-4: Many-to-many architecture


• One AR System server can support any number of AR System clients running on any mix of platforms.
• An AR System client can be connected to multiple AR System servers simultaneously.
• AR System servers can keep each other synchronized using the Distributed Server Option.
AR System clients
Distributed Server Option
AR System server
Tokyo, Japan
AR System server
Bracknell, England

AR System components

AR System applications address business processes such as problem management or asset tracking. Similar to Microsoft Excel templates, which are the spreadsheet forms and the underlying formulas, AR System applications are the forms and the related workflow definitions.

When an application is created using BMC Remedy Developer Studio, it is built with objects. There are five essential classes of objects that can be linked together to make applications. The first two provide the basic user interface pieces:

- **Forms**—Display information in fields. (Adding fields to a form causes the AR System server to create columns in a database table.) Each *data field* on a form has a set of properties that define the size of the field, the type of data that the field stores, and any access permissions. There are also several fields that don't contain data but instead organize data or improve the appearance of the screen: active link control fields (buttons and hotlinks), table fields, trim fields, and panel fields. Fields from existing forms can be combined into join forms.
- **Menus**—Are attached to fields on forms as fill-in aids. They can provide suggestions for entering data into a field, or can be mandated as the only possible choices. Menus can be statically defined, dynamically built by querying other AR System database tables, read from text files written by other applications, or created from SQL queries to external databases.

The other three object classes provide the workflow definitions:

- **Filters**—Action-based workflow definitions that are executed on the AR System server.
- **Active links**—Action-based workflow definitions that are executed on the client.
- **Escalations**—Time-based workflow definitions that are executed on the server.

"Action-based" means that the workflow definition is evaluated when there is a change of state of some data or some specific action is initiated. For example, a filter could be defined so that whenever a new trouble ticket record is submitted to the server with a priority of "high" or "critical," a notification message is sent to a support manager. Submitting the new record is the action. The requirement that the record have a certain priority is the "qualification" on the filter. Some of the other actions that can trigger filters are updating records, deleting records, and retrieving records. Active links can be triggered by many additional mechanisms, including clicking buttons on a form, entering carriage returns in fields, making menu selections, opening or closing windows, and others.

"Time-based" means that the workflow definition is evaluated based on a time parameter. These can be either absolute time, such as "every day at 2:00 p.m.," or a time interval, such as "once every 37 minutes."

Workflow definitions can be "qualified" as just described. A qualification is a logical expression that is evaluated when the workflow definition is triggered. For example, the qualification to enforce the priority requirements in the previous example is something like this:

```
$Priority$ = “High” OR $Priority$ = “Critical”
```

This says that the value in the Priority field of the particular record must match either the value “High” or “Critical.” Qualifications can be combinations of algebraic and Boolean structures.

If a workflow definition is triggered and the qualification is met, one or more actions can be taken by AR System. Actions include:

- Copying data from other forms or sending data to other forms
- Sending unscrewing messages to users or sending notifications using email, BMC Remedy Alert, or other methods
- Enabling or disabling fields or changing menus associated with fields
- Making DDE or OLE connections to other applications
- Running an external process
- Managing guides
- Managing dialog boxes, which are fields that are displayed to users who are filling out forms
- Error checking
- Logging information to a file, usually to maintain an audit trail
- Running an SQL command

Another object, called a guide, is a group of active links or filters that can assist users in accomplishing specific tasks. For example, a guide could open a business card form and then display instructions to users as they tab through the fields. Guides can also be used to group together reusable sets of workflow that can be referenced by several forms or actions in your applications.

Security and access control

Throughout AR System, keeping information secure is a major consideration. AR System implements a multitier approach to control access at the following points:

- Server level
- Form level (or database table level)
- Request level (or row level)
- Field level (or column level)
- Active link or guide level

When users start an AR System client, they must enter a user name and a password, which are checked against every AR System server with which the client is trying to connect. After a connection is made, each request that goes between the client and the server has the current user name and password checked to verify that the values are *still* valid.

In addition to having a unique user name and password on a server, every user is a member of zero or more *groups*. Groups are defined and maintained with the Group form, where each record is a different group definition. For example, there might be groups defined for First-Level Support, Back-Line Support, and Support Management. Groups are used to control information access to forms, requests, fields, and active links / guides. As a practical matter, most users are likely to belong to the Public group.

You could use group access to forms so that a particular form is visible to users in the Support Management group, but not to users in the First-Level Support and Back-Line Support groups.

For a particular form, an administrator can determine that certain requests are accessible only by members of one group and that other requests are accessible by members of a different group.

In addition, every field on a form has access control. You set field permissions when you define the field properties in BMC Remedy Developer Studio. Each field can have a list of groups that can view the field and the data entered into it. Some or all of the groups with View permission might also have “change” access so that they can enter and modify the data. If a user opens a form on his or her workstation and the groups he or she is a part of do not have View access to some of the fields, those fields are not displayed on the form. A field can also be visible to users or hidden so that it is accessible only through workflow.

Finally, each active link and active link guide has its access control assigned when it is created. A user who has access to an active link does not automatically have access to the field associated with it. Similarly, a user who has access to a guide does not automatically have access to the active links in the guide.

Access control in AR System is additive. That is, each user starts out with no access permissions; administrators then add permissions as needed. In this way, AR System implements strict access control. Administrators must make a conscious decision to grant access to specific groups on a case-by-case basis. However, if desired, the default permissions can be changed.

Only AR System administrators or subadministrators can modify security parameters.

Chapter

3 Integration considerations

Several mechanisms can be used to integrate AR System with another application. This section discusses the issues to consider when planning an integration project.

The following topics are provided:

- Where to integrate (page 36)
- Multiplatform issues (page 37)
- Choosing an implementation method (page 38)
- Integration technologies (page 38)

Where to integrate

The three options for integration points with AR System are the client, the server, and the database server. The choice depends on the nature of the integration and whether user interaction is involved.

AR System client

- **AR System to third-party application**—Integration with the AR System client typically involves taking data from a form and passing it to another application where the user can then perform some additional function. Integration can also simply consist of launching another application that reads data from the AR System database. In general, client integration assumes that the user will access the other application to some extent. Most instances are real-time, where a user is involved right now.
- **Third-Party application to AR System**—Often, a third-party application launches BMC Remedy User and directs it to display specific data. For example, a network management system might have a graphical map of the network devices. Selecting a device on the map and choosing a "List Open Tickets" from a menu could cause BMC Remedy User to be triggered with the ID of the selected device passed as a parameter, and generate a results list of all of the open trouble tickets for the device. This way, a network technician can quickly see all of the outstanding problems for a device, but does not need to know the details of starting AR System and issuing queries.

AR System server

Integration with the AR System server generally implies data sharing or transfer, either to or from the server. The integration might involve workflow that triggers secondary actions. Sometimes, the server initiates the interaction. For example, a filter is triggered that uses a Run Process action to call a pager application to send a notification to a technician. In other instances, a third-party application might submit new requests to the server or query for the status of existing requests. For example, a system management agent running on a PC might discover the addition of a new sound card. The agent sends a message to a (remote) management application that, in turn, submits a new request to an asset application in AR System. AR System users are not directly aware that a new request has been created, but the next time someone generates an asset report, the new information is included.

Database integration

The following three modes of integration involve the database directly:

- A third-party application reading the AR System database
- AR System reading an external database
- AR System writing to an external database

The first two modes, which involve reading databases, are relatively straightforward. Any application that can issue SQL commands and which has the appropriate permissions can read the data in the AR System tables. In a similar manner, AR System workflow actions can execute SQL read commands and scripts that query external database tables and retrieve information. For more information, see the *Database Reference*.

The third mode, having AR System write data to an external database table, can be accomplished using Direct SQL. Another method is to create AR System workflow that executes an SQL command script, passing any AR System data as parameters to the script.

In addition, View and Vendor forms are available to provide access to external databases in AR System forms.

Having a third-party application write data to an AR System table is not supported. The AR System server maintains the relationships among the tables in the AR System database. If a third-party application attempts to add data and does not maintain these relationships, the entire database can become corrupted.

Multiplatform issues

A major consideration for every integration implementation is the range of platforms that are involved. For example, on the AR System client side, some functions that work in BMC Remedy User are not available in a browser environment. For example, one of the active link actions is to issue a Windows DDE command. This action is ignored if executed from a mid tier client.

AR System clients are available for Windows and on all major platforms through the Web using the BMC Remedy Mid Tier. In some cases, integration at the client level is unique for the different platforms. The AR System workflow qualifications include functions to test for the different platforms. For example, the `$CLIENT-TYPE$` function identifies whether the client is BMC Remedy User or a browser. Multiple workflow definitions can be triggered in parallel, and the one appropriate for the platform is executed. In some cases, you can avoid the client functionality issue by running a process on the server from client workflow. Because the application executes on the AR System server’s platform and operating system, it doesn’t matter which client platform triggered the workflow.

Similarly, different AR System server platforms might require adjustments to integration implementations.

Choosing an implementation method

The following section outlines some methods by which you can implement integration with AR System.

- How quickly do you want to have the integration working?

  Some options are easy to get running for demonstration purposes, but have drawbacks for production deployment. The more complex the integration, the more time is required to implement it.

- How “robust” does the integration need to be? How heavy will the usage be, and are there any data throughput requirements?

  For an integration that will be used infrequently, some of the methods are simple to implement. If the integration involves moving a large amount of information, other methods might require more effort but produce better results.

- Will users be able to modify the integration? How will it be supported?

  Some of the methods do not lend themselves to user modification. Others are easily modifiable, but might be more difficult to support if changes are made.

Integration technologies

A basic design philosophy of AR System is that it is almost always used in conjunction with other tools and products to create an integrated solution. AR System is designed to have many integration points, making it easy to combine with your other solutions.

This table lists the technologies available for integrating with AR System:

Method	Description	Page
C API (Application Programming Interface)	The AR System API on the server is the most technically complex method. It requires knowledge of C programming and building executables. However, it provides access to all AR System server functionality for tightly linked, high-performance integration.	page 41
Java API	The AR System Java API is a collection of Java classes that provide AR System C API functionality in a Java development environment. Using this abstraction layer allows developers to quickly build enhanced applications for the web.	page 49
Web Services	This integration technology (XML, WSDL, UDDI, and SOAP) allows you to build distributed applications without programming: • Use the Set Fields workflow action and a Web Services object to “consume” third-party web services in AR System applications. • Use AR System to create and “publish” a Web Services object.	page 61

Method	Description	Page
AR System Plug-Ins	AR System clients perform data operations on external systems through the AR System server, plug-in service, and plug-in related APIs. The plug-in service extends the AR System server to integrate with external data sources. The AR System server connects to the plug-in service, which activates the proper plug-in when a transaction is made.	page 105
Data Visualization Field	The data visualization field provides a framework and services for BMC Remedy Mid Tier-based graphing solutions. It provides an efficient way to add graphical elements (such as flashboards) to AR System forms.	page 167
AREA Plug-Ins (AR System External Authentication)	BMC Remedy provides a special API that allows user logins to be validated from a data source outside the AR System database. This API is called the AR System External Authentication (AREA) API.	page 124 and page 127
Filter API Plug-Ins	The filter API enables you to use filters to permit other applications to make calls back into AR System.	page 105
ARDBC Plug-Ins (AR System Database Connectivity)	AR System Database Connectivity enables you to access and manipulate data that is not stored in AR System. Using the ARDBC API, you can create plug-ins used by AR System server to manage data. These plug-ins are loaded at run time and implement calls that are analogous to the set, get, create, delete, and get-list API calls for entries in a form.	page 105 and page 135
Vendor Forms	Vendor forms allow AR System to present data from external sources as entries in an AR System form. Vendor forms require you to have an ARDBC plug-in installed and configured.	page 183
View Forms	View forms allow direct read-and-write access to data in database tables that are not owned by AR System. This allows direct access to these tables, as if they were owned by AR System, without programming, database replication, or synchronization.	page 187
SQL Database Access	Third-party tools with appropriate permissions can access any information in the AR System database. In addition, AR System workflow can query other databases.	page 197
ODBC Access	ODBC (Open DataBase Connectivity) is a standard database access method developed by Microsoft. Using the BMC Remedy ODBC driver, any client capable of accessing ODBC can have read-only access to AR System forms. Reporting is a common use of ODBC.	page 201
BMC Atrium Integration Engine (AIE)	The BMC Atrium Integration Engine mediates between the AR System server and vendor applications such as SAP®, Oracle, and other applications and databases for which adapters are developed. Adapters can come from BMC Remedy, from partners, or from customers.	page 221
Command Line Interface (CLI)	A Command Line Interface (CLI) is available with most AR System clients. This enables a client to be started and passed a set of parameters so that either it is in a specific state and displays information or it completes a process and exits with no user interface displayed.	page 233
XML Import and Export	AR System can export and import object definitions in XML. AR System clients can convert AR System objects to XML and vice versa without making calls to the AR System server. When the server exports the file in XML format, it adds a header required to make it a valid XML document. This same header is required for the server to import an XML file correctly. Otherwise, the file is assumed to be in the standard AR System definition format.	page 233

Method	Description	Page
Running External Applications (Run Process)	One of the actions available in AR System workflow is the Run Process action. AR System can use the command line interfaces of other applications to start those applications and pass them initial data. In some cases, the third-party application is simply started, while in others AR System waits for a response.	page 259
OLE Automation	BMC Remedy User supports OLE Automation. It can be both an Automation server and Automation client. This enables AR System to send commands or data to other applications and to receive commands or data from other applications.	page 269
Dynamic Data Exchange (DDE)	BMC Remedy User supports DDE. It can be both a DDE client and server. This enables AR System to pass data to other Windows applications and to be started “in context” by other applications.	page 285
SNMP	You can use SNMP to manage complex networks through SNMP-compliant management consoles and monitor network devices.	page 307
Licensing Applications	Authorized integration system vendors (ISVs) can make their applications licensable at the application and user levels.	page 323

Chapter

4 AR System C API

The AR System C API provides a common interface client programs, including AR System clients, can use to communicate with AR System server. The C API is defined by a set of functions and data structures, and is implemented as a C library that you can link into your own programs. This section provides an overview of the AR System C API and how you can use it to integrate AR System with a third-party product.

The following topics are provided:

- Overview (page 42)
- Understanding the AR System API (page 43)
- Program structure (page 44)
- Multithreaded API clients (page 45)
- Using the API for integration (page 46)
- Issues and considerations (page 48)

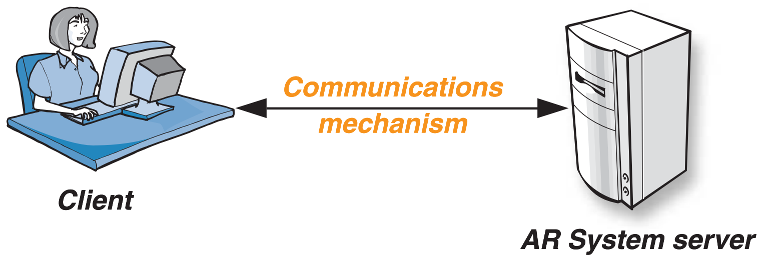
Overview

A client-server application model is a combination of the following items:

- A software component with a user interface on a client machine
- A software component on a server that processes and stores information
- A communications mechanism between the client and the server.

The user works at the client machine, which is "serviced" by the server.

Figure 4-1: Simple client-server model



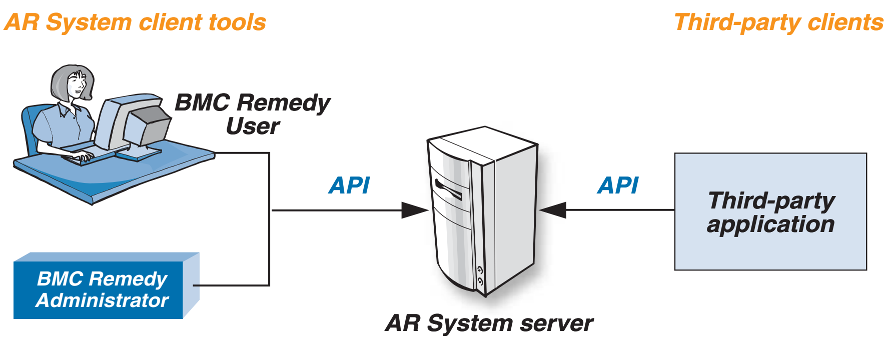
Typically, the client and server components are on different computer systems, and the communications mechanism is based on some form of networking technology, but it is possible to have both the client and server on the same host simply communicating between processes.

To make the development of client-server applications easier, an application programming interface (API) is used. Typically, the server is designed first, and its functionality is structured into a set of "commands." These commands are programmed into an API, which becomes the "language" for interacting with the server. When a client wants the server to do something, it makes a request through the API. The API handles all of the communications and operating system functions, gets the server to perform the task, and returns the appropriate information to the client. When the client software is developed, the programmer does not need to know any of the details of the server environment or communications mechanisms.

The AR System server has a fully defined API that is common to all server platforms, both Windows and the UNIX platforms. The AR System client tools all use this API for interaction with the servers. They all speak the same language and are completely interchangeable. A client on any platform can work with a server on any platform. As long as a client can connect to an AR System server, it can communicate all its requests and receive the replies using this common language. The client does not need to know on which platform the server is running. It also does not need to know about other clients that are using the same server.

The AR System API is defined by a strict set of programming functions and data structures, which are documented in the *C API Reference*. The API is implemented as a C library and associated files that can be linked into your programs. Third-party programs linked to the AR System API library become clients to an AR System server.

Figure 4-2: AR System API



Understanding the AR System API

The AR System C API and Java API are shipped on the product distribution media as installation options. There is no additional license fee or charge for using the APIs to develop custom client applications, nor is there a license fee or charge for redistributing or selling such custom clients.

The AR System components were built with these APIs. Therefore, any functionality available with these clients can be replicated in another client program.

The functions provided in the API libraries give the programmer control over all the following AR System components:

- **Requests**—Records in the database. Analogous to a row in a database table. Also known as entries.
- **Forms**—Objects used to query, modify, and submit records. Also known as schemas.
- **Views**—Various form (schema) layouts. Also known as VUIs.
- **Fields**—Objects in which data is entered on a form. Also used to enhance the appearance of a form. Analogous to a column in a database table.
- **Menus**—Objects that can be attached to one or more fields often used to improve ease-of-use when selecting a value for a field.
- **Filters**—Server-side, action-based workflow rules.
- **Escalations**—Server-side, time-based workflow rules.
- **Active links**—Client-side, action-based, and time-based workflow rules.

- **Containers**—Generic lists of references that are used to define guides and applications.
- **Support files**—Files that clients can retrieve. Commonly used for reports but can be any file on the server.

The API functions allow five actions to be performed on these objects:

- **Create**—Create an instance of the object.
- **Delete**—Delete an existing instance of the object.
- **Get**—Get details about an existing instance of the object.
- **Set**—Set (update) an existing instance of the object.
- **GetList**—Search and get a list of matching instances of the object.

Additional functions are available for Data Import, Data Export, User Administration, and Connection Control. API functions that deal exclusively with BMC Remedy Alert are also available.

Program structure

The AR System C API programs consist of the following four basic sections:

- **Startup/Initialization**—Call `ARInitialization` to perform server- and network-specific initialization operations for connecting to the AR System servers (required).
- **System Work**—Call one or more C API functions to perform the specific work of your program.
- **Free Allocated Memory**—Call one or more of the `FreeAR` functions (or use the free function) to free all allocated memory associated with a specific data structure. For more information, see the *C API Reference*.
- **Shutdown/Cleanup**—Call `ARTermination` to perform environment-specific cleanup routines and disconnect from the AR System servers (required).

  If you use floating licenses and do not disconnect from the server, your license token is unavailable for other users for the defined time-out interval.

Multithreaded API clients

The AR System API supports multithreaded clients through the use of sessions. Each session maintains its own state information, enabling simultaneous operations against AR System servers. This feature enables more sophisticated client programs to perform multiple operations simultaneously against the same or different servers. You establish a session with a call to `ARInitialization` and terminate it with a call to `ARTermination`. The session identifier returned in the control record from an `ARInitialization` call must be in the control record for all subsequent API function calls intended to operate in that session. Operations for a session are not restricted to a single thread; however, each session can be active on only one thread at a time.

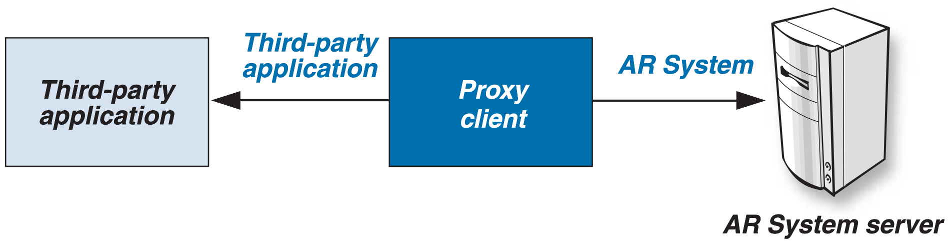
Following is an example of a generic API program source file.

```
#include <stdlib.h>
#include <string.h>
#include <ar.h>
#include <arextern.h>
#include <arfree.h>

int main( int argc, char* argv[] )
{
   ARStatusList status;

   /* read command line arguments */
   if (argc < 6) exit(-1);
   char* server = argv[1];
   char* user   = argv[2];
   char* pass   = argv[3];
   char* schema = argv[4];
   char* eid    = argv[5];

   /* set control parameters (server, user, etc) */
   ARControlStruct control;
   control.cacheId          = 0;
   control.sessionId        = 0;
   strcpy(control.user,     user);
   strcpy(control.password, pass);
   strcpy(control.language, "");
   strcpy(control.server,   server);

   /* Remedy Startup */
   if (ARInitialization(&control, &status) >= AR_RETURN_ERROR) {
      printf("\n **** initialization error ****\n");
      exit(-1);
   }
   FreeARStatusList(&status, FALSE);
   /* Verify user/password/server */
   if (ARVerifyUser(&control, NULL,NULL,NULL,&status) >=
AR_RETURN_ERROR) {
      printf("\n **** verification failed ****\n");
      exit(-1);
   }

   /* Set the entryid of the record we want */
   AREntryIdList entryId;
   entryId.numItems    = 1;
   entryId.entryIdList = (AREntryIdType *) calloc(1,
sizeof(AREntryIdType));
   strncpy( entryId.entryIdList[0], eid, AR_MAX_ENTRYID_SIZE );
   entryId.entryIdList[0][AR_MAX_ENTRYID_SIZE] = '\0';

   /* Get the entry */
   ARFieldValueList fieldValues;
   if( ARGetEntry(&control,
         schema, &entryId, NULL, &fieldValues,
         &status) >= AR_RETURN_ERROR) {
      printf("\n **** get entry error ****\n");
      exit(-1);
   }


   /* Clean up */
   FreeARStatusList(&status, FALSE);
   FreeARFieldValueList(&fieldValues, FALSE);
   FreeAREntryIdList(&entryId, FALSE);

   /* Terminate */
   (void) ARTermination(&control, &status);
   FreeARStatusList(&status, FALSE);

   return 0;
}
```

Using the API for integration

Typically, the AR System API is not linked directly into a third-party application. Instead, a separate program is created that interfaces with the AR System server using the AR System API on one side and with the third-party application using its native interface on the other side. This interface program acts as a proxy between the two applications, functioning as a client to both sides.

Figure 4-3: Using APIs to link applications



The proxy client that links AR System with another application does not need to provide all of the features seen in AR System clients such as BMC Remedy User. Only the functions necessary for useful integration need be implemented. A proxy client could run as a background process with no user interfaces. For example, a proxy client could be created to monitor a log file that a third-party application updates. Whenever new entries appear in the log file, the proxy client application could automatically submit new records into the AR System database, with no user interaction. Similarly, a proxy client could monitor the AR System database and periodically extract records and use them to create graphical reports or charts.

In addition to providing a means for custom clients to access the AR System server, the API can be used to integrate with existing AR System or legacy applications.

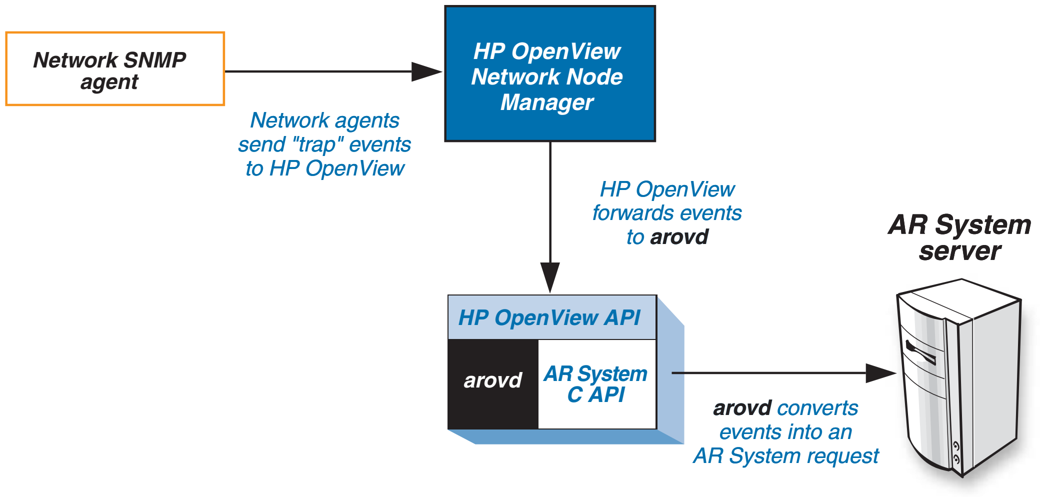
Example: Network management platform integration accessories

BMC Remedy makes available a set of accessories that provide integration with network management platforms such as HP OpenView Network Node Manager, IBM® NetView® for AIX®, and Oracle Solstice Domain Manager. The major portion of the integration consists of a set of proxy client applications that take selected events (or alarms or alerts) identified by the management platforms and create trouble ticket records in an AR System server. There is a unique proxy client application for each management platform.

The management platform applications run on UNIX hosts. The proxy clients run as background processes on these hosts. Each proxy client implements the management platform API to get the event messages (on one side) and the AR System API (on the other side) to send the information to an AR System server and create a trouble ticket. Usually, a proxy client communicates with a management platform locally within the host system and communicates with an AR System server remotely across a network. However, if AR System and the management platform are running on the same host, everything can be implemented locally.

For example, for HP OpenView Network Node Manager, the proxy client is called `arovd` (AR System OpenView daemon). It was built using the HP OpenView Event API, so that it could register itself with the HP OpenView system and receive events. It was also built with the AR System API so that events of interest could be translated and stored as records in the AR System database.

Figure 4-4: Integration of AR System with HP OpenView Network Node Manager


Network SNMP agent
HP OpenView Network Node Manager
Network agents send "trap" events to HP OpenView
HP OpenView forwards events to arovd
AR System server
HP OpenView API
arovd
AR System C API
arovd converts events into an AR System request

Issues and considerations

Keep the following points in mind when using the AR System API:

- Using the AR System API requires expertise in C programming. It assumes a technical background and familiarity with the use of compilers.
- The AR System server is backward compatible, supporting all requests from applications that use the API libraries back to AR System 3.2. If you continue to link to one of these libraries, you do not need to make any changes to continue running your existing programs against newer servers. If you link to a newer version of the AR System API libraries, you will probably need to make changes to your programs. The main program structure and processing, however, need not change.

  For information about new and changed calls in the API libraries, see the AR System *Release Notes* for versions newer than your integration.

  For information about which API calls have been updated or replaced, see the *C API Reference.*

Chapter

5 Java API

The AR System Java API is an application programming interface to integrate with AR System. This chapter includes the information you need to start developing your AR System Java client. See the online documentation for details of the API.

The following topics are provided:

- Overview (page 50)
- Installed files (page 50)
- Run-time configuration (page 52)
- Programming model (page 52)
- Programming with the Java API (page 54)
- Sample (page 55)

Overview

The Java API is a collection of classes, interfaces, and relationships that provide full client functionality like the AR System C API in a style consistent with typical Java programming techniques. Like the C API, the Java API is forward and backward compatible with other versions of AR System.

Installed files

To install the Java API, select the API server component when you install the AR System server. See the *Installation Guide* for detailed information.

The Java API files are typically installed in the following directories:

- **UNIX**—`/usr/ar/`*`serverName`*`/api`
- **Windows**—`C:\Program Files\BMC Software\ARSystem\`*`serverName`*`\Arserver\api`

The C API files are also in this directory. To verify that the Java API is installed, check that files listed in the next section are present.

Contents of the AR System Java API installation

The following table lists the components of the Java API installation.

Directory	Component	Description

`JavaDriver`	`build.xml`
	`CommandProcessor.java`
	`Commands.java`
	`ImageExtractor.java`
	`InputFile.java`
	`InputReader.java`
	`JavaDriver.java`
	`LocaleCharSet.java`
	`OutputWriter.java`
	`PerfJavaDriver.java`
	`PerfThreadControlBlock.java`
	`RandomNumberThread.java`
	`RowIterator.java`
	`SyncObject.java`
	`ThreadControlBlock.java`
	`ThreadStartInfo.java`
	`WFD.java`
	`WfdCommands.java`
	`WfdOutputWriter.java`
		Sample Java code file that shows examples of using the Java API

Directory	Component	Description
lib (This directory also contains the required C API library files.)	apache_axis_license.txt apache_crimson_license.txt apache_log4j_license.txt arapi76.dll arapi76.jar arapi76.lib ardoc76.jar arjni76.dll arrpc76.dll arsys_sample.xml arutil76.jar arutiljni76.dll arutl76.dll arxmlutil76.dll arxmlutil76.lib axis.jar commons-discovery-0.2.jar commons-logging-1.0.4.jar javadriver.bat javadriver.jar javawfd.bat jaxrpc.jar jlicapi76.dll jlicapi76.jar log4j-1.2.14.jar log4j-1.2.8.jar log4j.properties log4j.xml saaj.jar websvc76.jar wsdl4j-1.5.1.jar xercesImpl.jar xmlParserAPIs.jar	API and supporting JAR and DLL files with the Javadoc™-generated HTML documentation of the API, XML configuration files, and batch files for the Java driver and the workflow debugger.

Run-time configuration

To run Java API programs, make sure your execution environment includes:

- Java runtime environment™ (JRE™) 5 or later. (Java SE 5 is also know as Java 1.5.0.)
- Required JAR files in the `CLASSPATH` environment variable or the `java` command `-classpath` command-line parameter. Include all the JAR file in the `lib` directory listed in "Contents of the AR System Java API installation" on page 50.
- C API library files in the `lib` directory in the correct path:
  - **AIX, Linux®, and Solaris™**—`LD_LIBRARY_PATH`
  - **HP-UX**—`SHLIB_PATH`
  - **Windows**—`PATH`
- To override the default Java API configuration, create the `arsys_api.xml` configuration file and make sure it is in the `CLASSPATH` environment variable or the `java` command `-classpath` command-line parameter. See the sample configuration file, `arsys_sample.xml`, in the `lib` directory for descriptions, valid values, and default values for the API configuration options.

WARNING

Installing BMC Remedy Encryption—Premium Security changes any active Java installation. If you install Premium Security encryption on a system that is running a client written using the AR System Java API and later change the Java installation by installing a different version of the JRE or Java Development Kit (JDK™) or switching the system from one installed Java version to another, you must reinstall Premium Security encryption to make sure that the new or other Java installation has the changes that installation makes.

Java driver

Test your environment by running the AR System server and the `JavaDriver` program. This program illustrates the capabilities of the Java API. The `JavaDriver` program works almost exactly the same as the C driver program. For more information, see the *C API Reference*.

Programming model

Consistent with object-oriented design, the AR System Java API represents AR System server objects as Java objects. Classes are defined for forms, fields, menus, active link, filter, escalations, and all other objects in an AR System application. Entry objects represent entries (requests) so your Java client can manipulate AR System data as well as definitions.

The following paragraphs summarize a few frequently used classes in the API. For more information, see the Java API online documentation HTML files in the `ardoc76.jar` file listed in "Contents of the AR System Java API installation" on page 50. To access the Java API documentation, unzip the `ardoc76.jar` file to create a tree of directories, then open the `index.html` file to see an overview of the entire AR System Java API documentation with links to the details. (To unzip a JAR file, use a zip utility the Java `jar` executable, which is in the `bin` directory of the Java JRE is installation. For example, `jar -xvf ardoc76.jar`.)

ARServerUser

The `ARServerUser` object represents the connection between your Java client program and the AR System server. It include session information such as user name, password, and server. A typical program starts by creating an `ARServerUser` object with user name, password, server name, and the like. Using the `ARServerUser` instance methods, it logs in to the AR System server; creates, gets, searches for, updates, and deletes server objects; and log out. A call to a get method returns a server object; a call to a getList (search) method returns a list of object identities (for example, names for forms or IDs for fields); and a call to a getListObjects method returns a list of objects.

Server objects

The Java API includes classes for all server objects: `Form`, `Field`, `View`, `ActiveLink`, `Escalation`, `Filter`, `Container`, `Menu`, `Entry`, and `SupportFile`. The `Form`, `Field`, `Container`, and `Menu` classes have subclasses to represent specialized types, for example, `RegularForm`, `IntegerField`, `ApplicationContainer`, and `SqlMenu`.

The `ARServerUser` create methods create the server objects. The get and getListObjects methods return them. Each set methods takes a server object parameter that specifies which object the server should update. For example, the `ARServerUser` method `setField` takes a `Field` parameter.

Cloning objects

Objects in the AR System Java API are cloneable. The `clone()` method performs a deep copy of the object. A deep copy is a copy of an object that contains the complete encapsulated data of the original object, allowing it to be used independently of the original Java object. Of course, if the original Java object represents an AR System server object, the clone represent the same object.

Exception handling

Errors are modeled through the `ARException` class. All error messages that are returned by the server are thrown as an `ARException` in the Java API. Warnings and informational status messages are not treated as exceptions, but are available using the `getLastStatus` method of the `ARServerUser` class.

Programming with the Java API

▶ Follow these steps to write the program:

1. Make sure your programming environment is set up correctly. You need:
   - Java Development Kit (JDK) 5 or later. (Java SE 5 is also know as Java 1.5.0.)
   - The AR System Java API files. (See "Installed files" on page 50 for installation and verification.)
2. Create a Java project in your IDE.
3. Include the `arapi76.jar` and the other required JAR files in the AR System API `lib` directory in the class path.
4. Create a new class for the methods that call the Java API.
5. Import the `com.bmc.arsys.api` package and other packages you might use in your program. The API uses collection classes, so you are likely to need `java.util.ArrayList`, `java.util.List`, and `java.util.Map`.
6. Instantiate an `ARServerUser` object. Set the user name, password, server, and other connection attributes. If the program needs to interact with different servers or as different users, it can create more than one `ARServerUser` object.
7. Use the `ARServerUser login` method to open the connection to the server.
8. Perform the required operations using the `ARServerUser` methods and other API objects and methods to create, retrieve, update, and delete AR System system objects as needed and creating criteria objects and using server object methods as required.
9. Use the `ARServerUser logout` method to close the connection to the server.

Troubleshooting

Use these techniques to find errors in your Java API program:

- **Program fails to start**—Make sure all Java API and application dependant JAR files are in the class path.
- **Logging**—Configure logging using the `log4j.xml` file.

Sample

The following sample code illustrates how to use the Java API to create, modify, and query records in AR System:

```
package com.bmc.arsys.demo.samples;

import com.bmc.arsys.api.*;

import java.util.*;

public class JavaAPITest {
    private ARServerUser server;
    private String formName= "JavaAPITest";

    public JavaAPITest() {
        server = new ARServerUser();
        server.setServer("localhost");
        server.setUser("Demo");
        server.setPassword("");
    }

    public static void main(String[] args) {
        JavaAPITest test = new JavaAPITest();
        test.connect();
        test.createEntry("Demo","1","test1");
        test.createEntry("Demo","2","test2");
        String entryID = test.createEntry("Demo","3","test3");
        test.modifyEntry(entryID);
        test.queryEntrysByID(entryID);
        test.queryEntrysByQual(
            "( \'Create Date\' > \"1/1/2004 12:00:00 AM\" )");
        test.queryEntrysByQual("( \'Create Date\' > \"1/1/2010\"
)");
        test.cleanup();
    }

    // Connect the current user to the server.
    void connect() {
        System.out.println();
        System.out.println("Connecting to AR Server...");
        try {
            server.verifyUser();
        } catch (ARException e) {
            //This exception is triggered by a bad server,
password or,
            //if guest access is turned off, by an unknown
username.
            ARExceptionHandler(e, "Cannot verify user " +
                server.getUser() + ".");
            System.exit(1);
        }
        System.out.println("Connected to AR Server " +
            server.getServer());
    }

    // Create an entry in a form using the given field values.
    public String createEntry (String submitter, String status,
        String shortDesc) {
        String entryIdOut= "";
        try {
            Entry entry = new Entry();
            entry.put(Constants.AR_CORE_SUBMITTER, new
Value(submitter));
            entry.put(Constants.AR_CORE_STATUS,
                new Value(status, DataType.ENUM));
            entry.put(Constants.AR_CORE_SHORT_DESCRIPTION,
                new Value(shortDesc));
            entryIdOut = server.createEntry(formName, entry);
            System.out.println();
            System.out.println("Entry created. The id # is " +
                entryIdOut);
        } catch (ARException e) {
            ARExceptionHandler(e, "Cannot create the entry." );
        }
        return entryIdOut;
    }

    // Modify the short description field on the specified entry.
    void modifyEntry(String entryId) {
        try {
           Entry entry = server.getEntry(formName, entryId, null);
           entry.put(Constants.AR_CORE_SHORT_DESCRIPTION,
               new Value("Modified by JavaAPITest"));
           server.setEntry(formName, entryId, entry, null, 0);
           System.out.println();
           System.out.println("Entry #" + entryId +
               " modified successfully.");
        }
        catch(ARException e) {
            ARExceptionHandler(e,"Cannot modify the entry. ");
        }
    }

    // Retrive an entry by its entry ID and print out the number
of
    // fields in the entry. For each field in the entry, print out
the
    // value, and the field info (name, id and the type).
    void queryEntrysByID(String entryId) {
        System.out.println();
        System.out.println("Retrieving entry with entry ID#" +
entryId);
        try {
            Entry entry = server.getEntry(formName, entryId,
null);
            if( entry  == null ){
                System.out.println("No data found for ID#" +
entryId);
                return;
            } else
                System.out.println("Number of fields: " +
entry.size());

            // Retrieve all properties of fields in the entry.
            Set<Integer> fieldIds = entry.keySet();
            for (Integer fieldId : fieldIds){
                Field field = server.getField(formName,
                    fieldId.intValue());
                Value val = entry.get(fieldId);
                // Output field's name, value, ID, and type.
                System.out.print(field.getName().toString());
                System.out.print(": " + val);
                System.out.print(" , ID: " + field.getFieldID());
                System.out.print(" , Field type: " +
                    field.getDataType());
                // Handle DateTime value.
                if ( field instanceof DateTimeField ){
                    System.out.print(", DateTime value: ");
                    Timestamp callDateTimeTS =
(Timestamp)val.getValue();
                    if (callDateTimeTS != null)
                      System.out.print(callDateTimeTS.toDate());
                }
                System.out.println("");
            }
        } catch( ARException e ){
            ARExceptionHandler (e,
                "Problem while querying by entry ID.");
        }
    }

    // Retrieve entries from the form using the given
qualification. With
    // the returned entry set, print out the ID of each entry and
the
    // contents in its shortDescription field.
    void queryEntrysByQual(String qualStr) {
        System.out.println();
        System.out.println ("Retrieving entryies with
qualification " +
            qualStr);
        try {
            // Retrieve the detail info of all fields from the
form.
            List <Field> fields =
server.getListFieldObjects(formName);
            // Create the search qualifier.
            QualifierInfo qual =
server.parseQualification(qualStr,
                fields, null, Constants.AR_QUALCONTEXT_DEFAULT);
            int[] fieldIds = {2, 7, 8};
            OutputInteger nMatches = new OutputInteger();
            List<SortInfo> sortOrder = new ArrayList<SortInfo>();
            sortOrder.add(new SortInfo(2,
Constants.AR_SORT_DESCENDING));
            // Retrieve entries from the form using the given
            // qualification.
            List<Entry> entryList = server.getListEntryObjects(
                formName, qual, 0,
Constants.AR_NO_MAX_LIST_RETRIEVE,
                sortOrder, fieldIds, true, nMatches);

            System.out.println ("Query returned " + nMatches +
                " matches.");
            if( nMatches.intValue() > 0){
                // Print out the matches.
                System.out.println("Request Id         " +
                    "Short Description" );
                for( int i = 0; i < entryList.size(); i++ ){
                    System.out.println
(entryList.get(i).getEntryId() +
                        "     " +

entryList.get(i).get(Constants.AR_CORE_SHORT_DESCRIPTION));
                }
            }
        } catch( ARException e ) {
            ARExceptionHandler(e,
                "Problem while querying by qualifier. ");
        }
    }

    public void ARExceptionHandler(ARException e, String
errMessage){
        System.out.println(errMessage);
        printStatusList(server.getLastStatus());
        System.out.print("Stack Trace:");
        e.printStackTrace();
    }

    public void printStatusList(List<StatusInfo> statusList) {
        if (statusList == null || statusList.size()==0) {
            System.out.println("Status List is empty.");
            return;
        }
        System.out.print("Message type: ");
        switch(statusList.get(0).getMessageType())
        {
            case Constants.AR_RETURN_OK:
                System.out.println("Note");
                break;
            case Constants.AR_RETURN_WARNING:
                System.out.println("Warning");
```

```
                    break;
                case Constants.AR_RETURN_ERROR:
                    System.out.println("Error");
                    break;
                case Constants.AR_RETURN_FATAL:
                    System.out.println("Fatal Error");
                    break;
                default:
                    System.out.println("Unknown (" +
                        statusList.get(0).getMessageType() + ")");
                    break;
            }
            System.out.println("Status List:");
            for (int i=0; i < statusList.size(); i++) {

System.out.println(statusList.get(i).getMessageText());

System.out.println(statusList.get(i).getAppendedText());
            }
        }

        public void cleanup() {
            // Logout the user from the server. This releases the
resource
            // allocated on the server for the user.
            server.logout();
            System.out.println();
            System.out.println("User logged out.");
        }
}
```

Chapter

6 Web services

This chapter provides information about using web services with AR System. It describes how to publish AR System functionality as a web service and how to invoke external web services to exchange data between AR System and web service applications.

The following topics are provided:

- Overview of web services in AR System (page 62)
- Setting up the environment for web services (page 66)
- AR System web services architecture (page 69)
- Publishing a web service (page 72)
- Registering a web service (page 80)
- Consuming a web service (page 89)
- SOAP headers and authentication (page 95)
- Supported schema constructs and AR System web service limitations (page 101)

Overview of web services in AR System

Web services provide a simple, platform-agnostic method for application integration. By using standard messaging protocols and service definitions to support computer-to-computer interaction over a network, web services allow application integration regardless of the hardware or software platforms and independent of the programming languages in which the applications are written.

You *publish* a web service to make AR System functionality available over the web by creating a web service object, associated forms, and optional workflow. AR System developers can use a web service to make AR System operations, such as submit, modify, and query, available to other applications. Web services that are published in AR System can be very basic, such as creating a record in the AR System database, or more complex, such as processing a purchase order that spans multiple AR System forms.

You *consume* a private or public web service to obtain external information for use in an AR System application by creating a form and an associated filter that uses a Set Fields action.

Web service standards

AR System web services use most standard web service messaging and transport protocols, XML schema constructs, message types, and operation types.

Protocols

AR System uses standard web service protocols, including:

- HyperText Transfer Protocol (HTTP)—The standard communication protocol for exchanging information the Web.
- Extensible Markup Language (XML)—A markup language defined in the XML 1.0 Specification, used to encode documents and represent data structures by describing data types. XML facilitates sharing data across different hardware and software platforms.
- Simple Object Access Protocol (SOAP) —The standard messaging protocol for exchanging information with web services. It is based on HTTP and XML and provides the envelope format for transferring information and a set of rules for translating applications and platform-specific data types into XML.
- Web Services Description Language (WSDL)—An XML-based language used to define a web service, it’s operations, and how to access it. For an example of a WSDL file, see Figure 6-6 on page 75.

NOTE

The AR System web services implementation is based on SOAP 1.1 and WSDL 1.1 specifications from the World Wide Web Consortium (W3C). SOAP 1.2 is supported for consuming web services only.

- Universal Description Discovery and Integration (UDDI)—A specification used to provide directories of information about available web services. The BMC Atrium Web Services Registry uses this standard, and if it is installed, AR System developers can use it to register and locate web services. See “Registering a web service” on page 80.

For more information about web service standards and protocols, see the W3C website at `http://www.w3.org`.

IMPORTANT

The web services namespace format structure was changed between the AR System version 7.0.1.3 release and the AR System version 7.0.1.4 release. In the earlier release, the tags were prepended with `ns1`, and with the newer release they are prepended with `ns0`. Because of that change, some customer implemented web services that were created on a version before the change may not work after upgrading to a version after that change was implemented. For any web services that stop running after upgrading, the SOAP Input document needs to be regenerated using the `ns1` tag.

Operation types

Each web service has a list of operations. AR System supports four operation types: `Get`, `Create`, `Set`, and `Service`. You can rename, delete, and even create operations, but they must be one of these supported operation types. You can have multiple operations of the same type, or you can have no operations of a particular type.

By default, when you create a web service, it automatically has these five operations:

- `Get`
- `Create`
- `GetList` (of the type `Get`)
- `Set`
- `Service`

For the procedures to add and remove operations, see “Creating a web service” on page 72 and “Consuming a web service” on page 89. For details about using each operation type, see Appendix A, “Web service operation types.”

Each web service is associated with an XML Schema Definition (XSD file). The XML schema defines the global elements and complex types that are used in the field mappings associated with operations. For a basic web service using the default operation types, AR System populates the XML Schema automatically. You can also define your own XML schema or use an existing one.

For more information about working with the XML schema, see “XML editing” on page 351. For a list of supported and unsupported XML constructs, see “Supported schema constructs and AR System web service limitations” on page 101.

WSDL types

In general, web services use these messaging styles:

- **Remote Procedure Call (RPC-style)**—One application makes a function call to another application, passing arguments and receiving return values.
- **Document-style**—Applications exchange XML documents whose syntax are defined by an XML schema, for example a Purchase Order document.

These styles can be divided into *literal substyles* (messages are encoded according to the XML schema) or *encoded substyles* (messages are constructed according to SOAP encoding rules). This results in four WSDL styles:

- RPC-literal
- RPC-encoded
- Document-literal
- Document-encoded

NOTE

For publishing, AR System supports *only* document-literal web services. For consumption, AR System supports document-literal and RPC-encoded web services.

Predefined AR System web services

The following web services are installed with AR System:

- User web service
- Group web service
- Roles web service

You can use or customize these three web services to allow external applications to find, create, and update entries in the User, Group, and Roles forms.

These web services are also defined in the `SystemWebService.def` file, located in the `~InstallForm/en` directory.

Forms and field mappings for web services

For both publishing and consuming web services, you specify a *base form* that is associated with the web service object (publishing) or with a filter (consuming). The information exchanged in the web service action is set in this form or pushed through it to other forms or applications. For web services that involve multiple AR System forms, the base form is the master form.

For each operation in the web service, you define an *input mapping* and an *output mapping*. Mappings are essentially the input and output parameters of the web service. The mapping describes how the elements of the incoming and outgoing XML document are mapped to the fields in the form.

AR System provides default input and output mappings for each of the default operations.You can use the mappings that AR System automatically creates or you can customize them. You can map to a simple flat WSDL document or to a complex hierarchical document involving parent and child relationships.

For the procedures used to define input and output mappings, see "Creating a web service" on page 72 and "Consuming a web service" on page 89. For details about defining mappings see "Mapping to simple and complex documents" on page 340.

Basic and custom web services

A web service published in AR System can be a basic or custom web service.

A basic AR System web service has five operations: `Get`, `Create`, `GetList`, `Set`, and `Service`. For each field on the base form, BMC Remedy Developer Studio generates an XML-compliant element name and maps it to an input and output parameter. The XML-compliant element names are used in the WSDL file that Developer Studio also generates for the web service. An external client can then call this web service and create, modify, or get records from your form. For the procedure to create a basic web service, see "Creating a basic web service" on page 72.

To customize a web service, you can change some of the web service definitions:

- Rename the web service. The default name is the same as the base form name.
- Rename operations.
- Remove operations and add new ones.
- Remove fields and add new ones to the mapping.
- Include only some parts of special fields in your mapping. For example, for an attachment field with multiple parts such as `attachmentName`, `attachmentData`, and `attachmentOrigSize`, you can select only the parts that you require.
- Rename XML element names. BMC Remedy Developer Studio names the XML elements the same as the field names but removes any special characters and spaces to make the names compliant with XML naming conventions. You can choose any XML-compliant name to map to your fields. You can also import an XML document and use the existing XML names to map to your fields.
- Modify the XML structure. For example, group certain fields to make complex-types or SOAP-structures, or designate a field as an attribute rather than a subelement.
- Specify an external XML schema.

Creating web service clients

When you publish an AR System web service, a WSDL document that describes the web service is created in Developer Studio. A web service client must be created to interact with this web service. Many environments can be used to create web service clients. Most have tools that automatically generate code to invoke a web service, given the WSDL. Apache AXIS, a third-party library, is installed with the mid tier for this purpose, but you can use other web service frameworks as well.

Popular environments for writing web service clients include these:

- **Apache AXIS**

  In AXIS, run `WSDL2java` from the command line with the WSDL URL as a command-line parameter. The autogenerated code is a class that has methods that correspond exactly to the operations you created in the web service. Each method has input and output parameters corresponding to the mappings you created. To invoke the web service, instantiate the class and invoke a method with the correct parameters. For more information, see `http://ws.apache.org/axis/java/user-guide.html` and `http://ws.apache.org/axis2/1_2/quickstartguide.html`.

- **JAX-WS**

  In JAX-WS, run `wsimport` from the command line with the WSDL URL as a command-line parameter. The autogenerated code contains the necessary class to invoke any operation on the web service. For more information, see `https://jax-ws.dev.java.net/guide/`.

- **Microsoft.NET**

  In Microsoft.NET Visual Studio, autogenerate the invocation code by adding a web reference. When prompted for a URL, enter your WSDL URL. For more information, see `http://msdn2.microsoft.com/en-us/library/w3h45ebk.aspx`.

Setting up the environment for web services

To use web services, you must install the BMC Remedy AR System server with the Web Services installation option, which is a Java plug-in, along with the BMC Remedy Mid Tier. For information about the environment prerequisites for the Java plug-in server and the mid tier, see the *Installation Guide*.

You must also install BMC Remedy Developer Studio to create and manage AR System forms, web services, and workflow.

If you are using the BMC Atrium Web Services Registry, additional components are required. See “Registering a web service” on page 80.

Verifying the AR System server configuration for web services

AR System uses the Java plug-in server to consume web services. To do so, the following line must appear in your `ar.conf` (`ar.cfg`) file:

```
Server-Plugin-Alias: ARSYS.ARF.WEBSERVICE ARSYS.ARF.WEBSERVICE
serverName:portNumber
```

In addition, the XML definition for `ARSYS.ARF.WEBSERVICE` must appear in the Java plug-in server configuration file, `pluginsvr_config.xml`.

The installation program creates these entries when you select the Web Services plug-in for an AR System installation or upgrade. If you need to update the files manually, make sure to stop and restart the AR System server before attempting to use web services. Alternatively, you can restart only the plug-in server. For more information, see "Restarting the plug-in server using the Set Server Info command" on page 118.

For more information about the Java plug-in server, see "Plug-ins" on page 105.

Configuring for a proxy server

If you will be using web services with a proxy server, perform the procedures in this section.

The first procedure modifies the BMC Remedy Developer Studio initialization file to enable communication with the proxy server. The second procedure enables the Java plug-in server to communicate with the proxy server (for example, with filters that consume web services). For more information about the Java plug-in server, see "Plug-ins" on page 105.

AR System does not automatically retrieve proxy settings from your browser. However, if the browser is configured to use a proxy server, you might be able to use those settings in the procedures below.

NOTE

The web services plug-in no longer uses the Proxy Server Setting For Java VM field on the Connection Settings tab of the Server Information form.

▶ To configure BMC Remedy Developer Studio for use with a proxy server

1 Verify that BMC Remedy Developer Studio is not running.

2 Open *ARSystemInstallDir*`\DevStudio\devstudio.ini` in a text editor.

3 Add the following lines to the file:

```
-Dhttp.proxySet=true
-Dhttp.proxyHost=hostName
-Dhttp.proxyPort=portNumber
```

4 Save and close the `devstudio.ini` file.

▶ To configure the Java plug-in server for use with a proxy server

1 Stop the AR System server.

2 Open `armonitor.conf` (`armonitor.cfg`) in a text editor. Default locations are:

   UNIX—`/etc/arsystem/`*`serverName`*`/`
   Windows—*`ARSystemServerInstallDir`*`\Conf\`

3 Locate the Java plug-in server command.

4 In the Java command, add the following proxy information immediately before the `-classpath` specification:

```
-Dhttp.proxySet=true -Dhttp.proxyHost=hostName
-Dhttp.proxyPort=portNumber
```

   A typical Java command in `armonitor.cfg` might look like this:

```
"C:\Program Files\Java\jreVersion\bin\java" -Xmx512m
-Dhttp.proxySet=true -Dhttp.proxyHost=hostName
-Dhttp.proxyPort=portNumber -classpath "C:\Program Files\BMC
Software\ARSystem\pluginsvr;C:\Program Files\BMC
Software\ARSystem\pluginsvr\arpluginsvr75.jar"
com.bmc.arsys.pluginsvr.ARPluginServerMain -x serverName -i
"C:\Program Files\BMC Software\ARSystem" -m
```

5 Save and close `armonitor.conf` (`armonitor.cfg`).

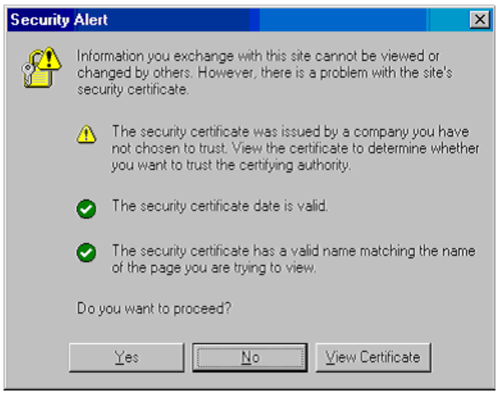
6 Restart the AR System server.

Accessing WSDL or web services over https

Make sure that a supported version of the Java Runtime Environment (JRE) is specified as the current JRE in the system where BMC Remedy Developer Studio and the web service filter plug-in are installed. Newer JRE installations include `JSSE.jar` and related settings that enable Java programs to communicate over https without manual changes in configuration files. If you must use an older version of the JRE, go to `http://www.oracle.com/technetwork/java/index.html` to learn more about JSSE-related `jar` files and how to make the related configuration changes.

You can check the validity of the certificate by using your browser. Browsers indicate errors and warnings in detail while communicating over https. For example, Figure 6-1 displays the warning that the certificate at the target server is not trusted by the local client.

Figure 6-1: Security alert for certificate that is not trusted



Security Alert

Information you exchange with this site cannot be viewed or changed by others. However, there is a problem with the site's security certificate.

The security certificate was issued by a company you have not chosen to trust. View the certificate to determine whether you want to trust the certifying authority.

The security certificate date is valid.

The security certificate has a valid name matching the name of the page you are trying to view.

Do you want to proceed?

Browsers allow you to continue if you ignore warnings or errors; however, AR System does not allow you to continue in case of errors or warnings. You can easily diagnose the problem with your browser, fix the root cause, and continue with AR System. To address the alert shown in Figure 6-1, you would update the certificate store to trust the specified company or certificate. See `http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html` or your local administrator to learn more about frequent issues with certificates.

AR System web services architecture

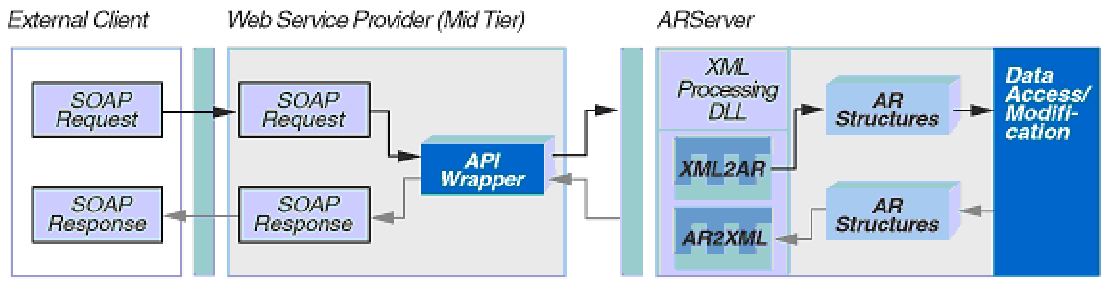
This section describes how information flows between AR System and client applications for web services published in AR System, and how information flows between AR System and an external web service consumed by an AR System application.

Information flow for web services published in AR System

When a client contacts an AR System web service, the interaction works as follows:

Step 1 The external client sends a Simple Object Access Protocol (SOAP) request to the mid tier. The URL for the service is either built into the application or is obtained from the web services registry at run time.

Figure 6-2: External web service client calling AR System (publishing)



Step 2 The mid tier extracts the web service name, the operation name, and the authentication information from the SOAP request packet. It retrieves the web service object corresponding to the web service name from the AR System server and searches the web service for details about the operation, such as the operation type (`Get`, `Create`, `Set`, or `Service`), the query string, and the input and output mappings. Then it expands the `XPATH` expressions in the query string, extracts the XML document from the SOAP request packet, and sends it to the AR System server along with the operation type, the input and output mappings, and the expanded query string.

Step 3 The AR System server parses the XML document using the mapping information and converts it into field data. The operation type determines how the data is treated:

- For `Get`, the AR System server ignores the input fields.
- For `Create`, the AR System server creates an entry with the input fields.
- For `Set`, the AR System server searches for an entry using the expanded query string and then modifies the data using the input fields.
- For `Service`, the AR System server sends the input fields to the Service Entry call.

For complex documents, the data is pushed into one or more forms. This action might trigger some filters.

Step 4 The AR System server also uses the mapping information to get the data from one or more records and generate an XML document. The operation type determines how the data is treated:

- For `Get`, the AR System server performs a query based on the query string.
- For `Create`, the AR System server reads the record that was created.
- For `Set`, the AR System server reads the record that was modified.
- For `Service`, the AR System server reads the output fields from the Service Entry call. This action might also trigger some filters.

Step 5 The XML document is returned to the mid tier.

Step 6 The mid tier packages the XML document as a SOAP response and returns it to the external client.

Information flow for consuming a web service in AR System

An external web service can be one created on another AR System server, or one based in some other environment, such as one that provides stock quotes, weather information, or currency exchange rates.

AR System communicates with the web service through the Web Service plug-in, using the third-party web service server (Apache AXIS) installed with the Java plug-in server.

The flow for consuming a web service in AR System is as follows:

Step 1 A filter process triggers a Set Fields filter action that sets fields using data from a web service.

Step 2 The filter uses the mapping information stored on the server to construct an XML document with data from the base form and the child form (if any).

Step 3 The AR System server sends the XML document to the web service plug-in.

Step 4 The web service plug-in receives the XML document, packages it into a SOAP request packet, and calls the external web service.

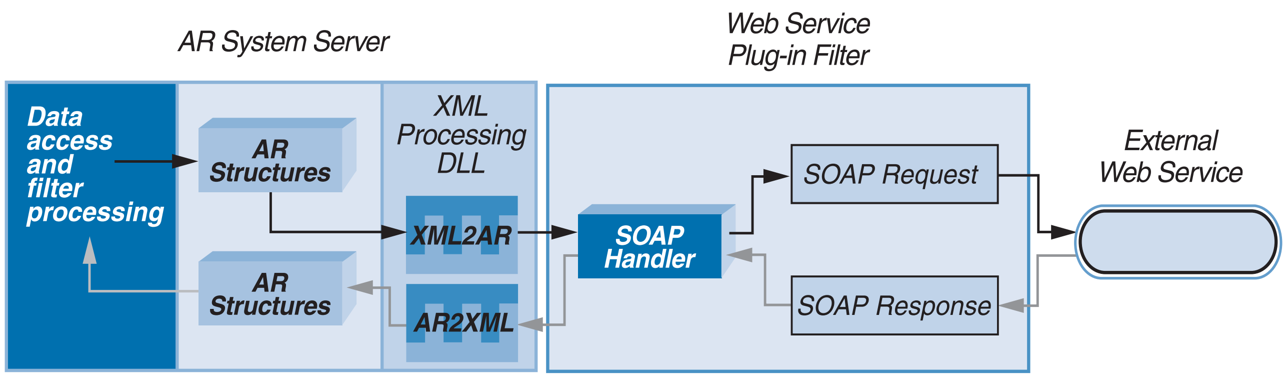
Step 5 The external web service replies with the SOAP response packet.

Step 6 The web service filter plug-in extracts an XML document from the SOAP packet and returns it to the AR System server.

Step 7 The AR System server receives the XML document and uses the mapping information to parse the document and push data into the current record and any child forms.

Step 8 The Set Fields filter action is finished.

Figure 6-3: Consuming an external web service with AR System



Publishing a web service

When you create and publish an AR System web service, you can make AR System operations available over the Internet or an intranet. This section describes how to create and configure the web service in AR System.

If the BMC Atrium Web Services Registry is installed, you can register the web service to allow applications to locate it at run time. For information about using the BMC Atrium Web Services Registry, see "Registering a web service" on page 80.

For information about creating the web client, see "Creating web service clients" on page 66 and your web application API documentation.

Creating a web service

You create and configure the web service in BMC Remedy Developer Studio. You can create a basic web service or a custom web service.

Each web service includes the following resources:

- A base form on which the web service operates. For web services use multiple AR System forms, the base form is the master form.
- An AR System web service object, which includes these definitions:
  - An associated XML Schema (`.xsd` file). Global elements and complex types defined in the XML schema can be used in the mappings for each operation. AR System provides a default schema, or you can select another one.
  - A list of Create, Get, GetList, Set, and Service operations. By default, an AR System web service is created each of these operation types. You can have more than one operation of the same type, or you can have no operations of a particular type.
  - A mapping for each operation that specifies how the elements of incoming and outgoing XML documents are mapped to fields in the base form.

Creating a basic web service

When you right-click a form in BMC Remedy Developer Studio and choose Create Web Service, AR System generates a basic web service. It is automatically associated with the form, and includes the default XML schema and the WSDL operations `Get`, `Create`, `GetList`, `Set`, and `Service`. Each of these operations includes a set of pre-defined input and output parameters using the fields on your base form. An external client can call this web service and create, modify, or get records from your form.

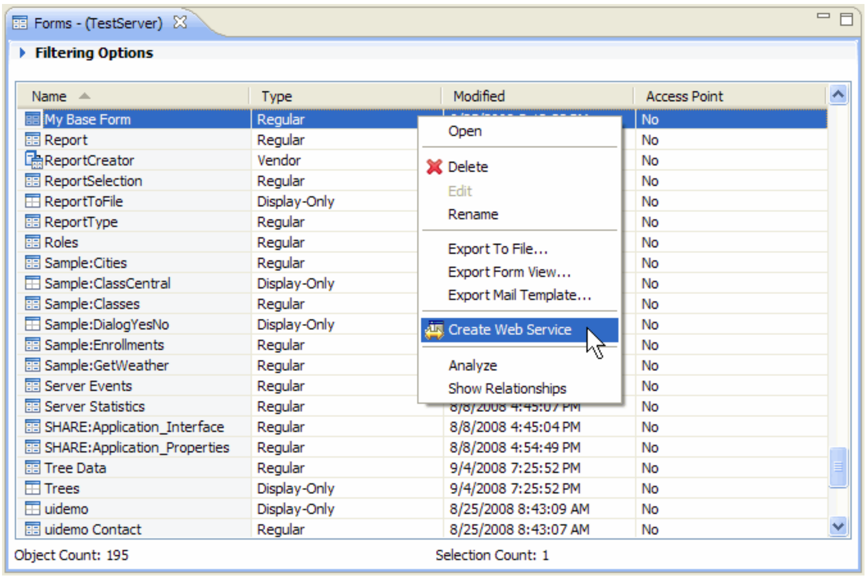
▶ To create a basic web service

1 Create a form to use as your base form, if you do not already have one.

2 In AR System Navigator, expand *serverName* > All Objects and double-click Forms.

3 Right-click your base form and choose Create Web Service.

Figure 6-4: Right-click the base form to create a web service



Name	Type	Modified	Access Point
My Base Form	Regular		No
Report	Regular		No
ReportCreator	Vendor		No
ReportSelection	Regular		No
ReportToFile	Display-Only		No
ReportType	Regular		No
Roles	Regular		No
Sample:Cities	Regular		No
Sample:ClassCentral	Display-Only		No
Sample:Classes	Regular		No
Sample:DialogYesNo	Display-Only		No
Sample:Enrollments	Regular		No
Sample:GetWeather	Regular		No
Server Events	Regular		No
Server Statistics	Regular	8/8/2008 4:45:07 PM	No
SHARE:Application_Interface	Regular	8/8/2008 4:45:04 PM	No
SHARE:Application_Properties	Regular	8/8/2008 4:54:49 PM	No
Tree Data	Regular	9/4/2008 7:25:52 PM	No
Trees	Display-Only	9/4/2008 7:25:52 PM	No
uidemo	Display-Only	8/25/2008 8:43:09 AM	No
uidemo Contact	Regular	8/25/2008 8:43:07 AM	No

Object Count: 195   Selection Count: 1

4 Right-click the General panel, then choose Expand All Panels.

These are the default settings for web services:

- **Form Name**—The name of the base form.
- **Label**, **Description**, and **XML Schema**—Blank.
- **XML Schema Source**—Embedded. The source type applies only if a schema file is specified in the XML Schema field.
- **Port Name**—The default value is Port.
- **Operations**—Default input and output mappings are automatically set for each WSDL operation.
- **Qualification**—An automatically generated qualification for `Get`, `Set`, and `GetList` operations. No qualification is needed for the `Create` or `Service` operations.
- **Start Record** and **Max Limit**—When `GetList` is selected, these fields are filled in. See "Setting the start record and the maximum limit" on page 337.

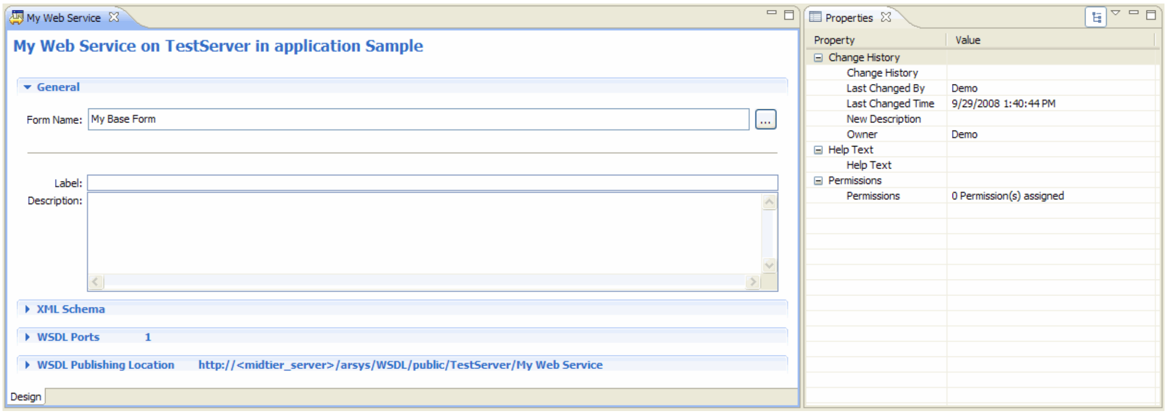
NOTE

The Start Record and Max Limit fields apply only to the `GetList` operation. For other `Get` operations, data entered in these fields is ignored.

- **Set Query Options**—The default setting is Set All Entries. See "Set operations with line items" on page 344.

- **Composite Options**—The default setting is Full. See “Set operations with line items” on page 344.

Figure 6-5: Web Services editor and Properties tab



My Web Service on TestServer in application Sample

General

Form Name: My Base Form

Label:

Description:

XML Schema

WSDL Ports 1

WSDL Publishing Location http://<midtier_server>/arsys/WSDL/public/TestServer/My Web Service

Property	Value
Change History	
Change History	
Last Changed By	Demo
Last Changed Time	9/29/2008 1:40:44 PM
New Description	
Owner	Demo
Help Text	
Help Text	
Permissions	
Permissions	0 Permission(s) assigned

5 In the Properties tab, set Permissions, Change History, and Help Text. The permissions are the same whether the web service is visible or hidden.

IMPORTANT

If you intend to publish your web service over an internet or intranet for general use, set the permissions to Public.

6 Click File > Save to save your web service.

The name of the web service should be descriptive and indicative of the web service’s function. The name should not be the same as any existing active link guide, filter guide, packing list, or application. The default web service name is name of the base form you chose when creating the service.

7 Click the WSDL Publishing Location panel.

A sample URL for your WSDL file is displayed in the “Specify mid-tier’s WSDL handler URL” field.

8 Adjust the URL for your configuration:

a Replace <*midtier_server*> with the name of the web server where the mid tier is running.

b Add `/public` or `/protected` after “WSDL,” depending on the permissions of the web service.

c Replace “Untitled Web Service” with the name of the web service.

For example, if the web service has public permissions, use this format:

```
http://midtierServer/arsys/WSDL/public/ARServer/webServiceName
```

If the web service does not have public permissions, use this format:

```
http://midtierServer/arsys/WSDL/protected/ARServer/
webServiceName
```

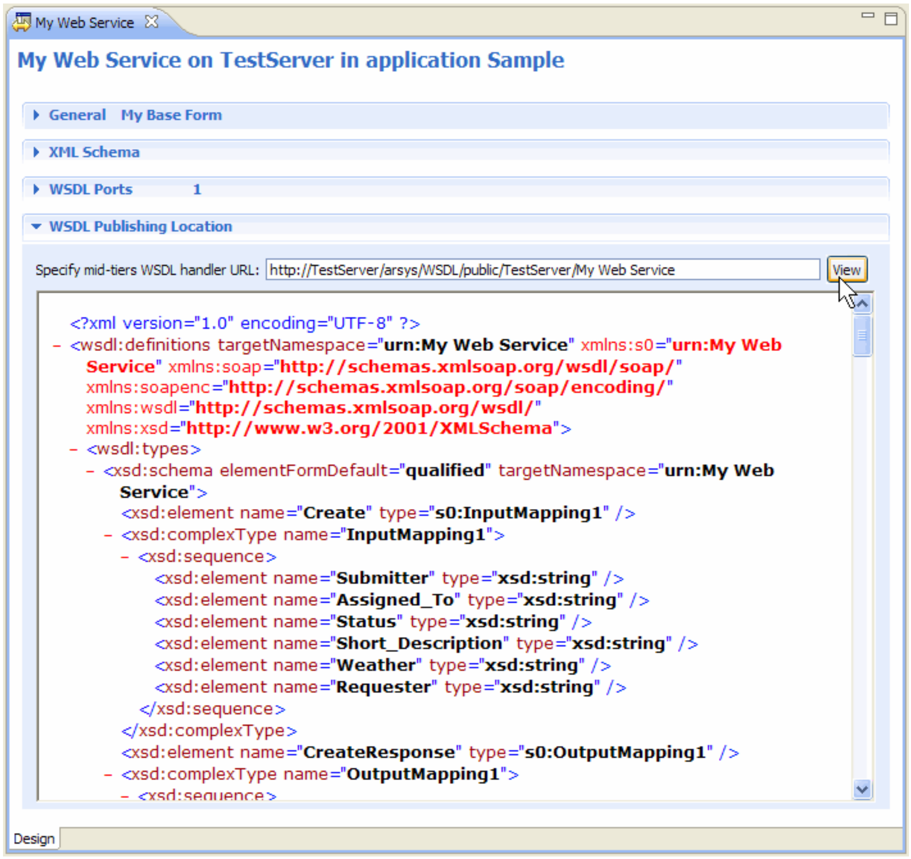
9 Click File > Save.

10 Click View to display your WSDL file in the panel.

This step also verifies that BMC Remedy Developer Studio can access the WSDL file.

To enable your clients to communicate with your web service, see “Creating web service clients” on page 66.

Figure 6-6: WSDL file displayed in WSDL panel of web services editor



Specify mid-tiers WSDL handler URL: http://TestServer/arsys/WSDL/public/TestServer/My Web Service

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="urn:My Web Service" xmlns:s0="urn:My Web
    Service" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <xsd:schema elementFormDefault="qualified" targetNamespace="urn:My Web
        Service">
        <xsd:element name="Create" type="s0:InputMapping1" />
      - <xsd:complexType name="InputMapping1">
        - <xsd:sequence>
            <xsd:element name="Submitter" type="xsd:string" />
            <xsd:element name="Assigned_To" type="xsd:string" />
            <xsd:element name="Status" type="xsd:string" />
            <xsd:element name="Short_Description" type="xsd:string" />
            <xsd:element name="Weather" type="xsd:string" />
            <xsd:element name="Requester" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="CreateResponse" type="s0:OutputMapping1" />
      - <xsd:complexType name="OutputMapping1">
        - <xsd:sequence>
```

Creating a custom web service

A custom web service is one in which you modify or add to the basic operation types or XML elements created by AR System. It might also use an existing XML schema (XSD file), rather than the default XSD created by AR System. To create a custom web service, follow the steps in this section.

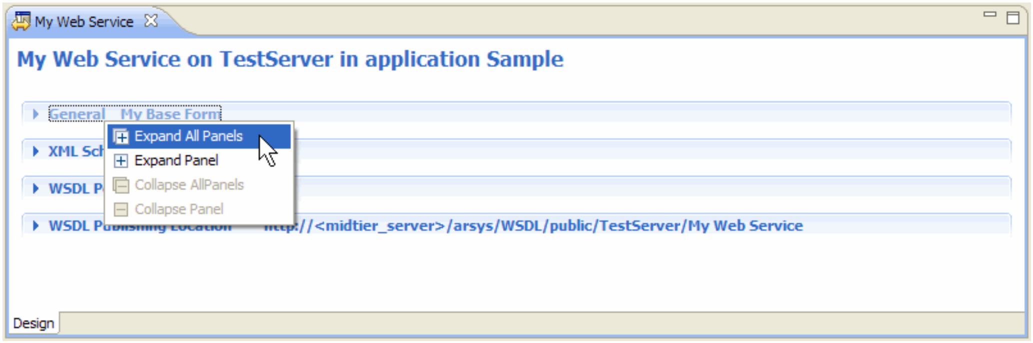
▶ To create a custom web service

1 In BMC Remedy Developer Studio, choose File > New > Web Service.

2 Select the server on which you want to create the web service, and click Finish.

3 Right-click the General panel, then choose Expand All Panels.

Figure 6-7: Web services editor with collapsed panels


My Web Service
My Web Service on TestServer in application Sample
General My Base Form
XML Sch
WSDL P
WSDL P http://<midtier_server>/arsys/WSDL/public/TestServer/My Web Service
Expand All Panels
Expand Panel
Collapse AllPanels
Collapse Panel
Design

4 Click the ellipsis button [...] next to the Form Name field, and choose a Base Form from the list of forms.

The web service operations function through the Base Form.

5 In the XML Schema field, perform one of these actions:

- Enter or browse to an external XML schema, and go to step 6.

  With an external XML schema, the elements or complex types in the file are used to map AR System form fields to operation parameters. See "Importing an external XML schema" on page 358.

- Leave the field blank, and go to step 8.

  BMC Remedy Developer Studio creates an XML schema with default element names.

6 Click Reload.

AR System verifies the XML schema and loads it into memory. You might see a "schema imported successfully" message or a message informing you that your existing XML elements and mappings will be lost.

7 Choose the XML Schema Source type in the drop-down list.

- If you entered a local file system path for your schema, select Embedded.

  AR System stores the entire XSD file and all other files that the XSD file includes or imports. The WSDL also has all the XSDs embedded in the types section. This is the default option.

- If you entered a network accessible path (http or ftp) for the schema, select Linked.

  AR System does not store the XSD files, and the WSDL does not embed the XSDs in the types section. Instead, it refers to them. Some WSDL parsing tools (early versions of Microsoft.NET and MSSOAP) do not support these kinds of WSDL.

NOTE

System-generated schemas are always embedded in the WSDL.

8 (optional) In the Label field, enter a label for the web service.

   Label names can have 80 or fewer characters, including spaces. The label is displayed in the Web Services list in the server window. If no label is specified, the web service name is used.

9 (optional) In the Description field, explain what the web service does and how it can be used so that callers of the web service can determine whether the web service includes the functionality they need.

10 Right-click the Port tab, choose Add Operation, and select an operation type from the list.

   The default operations are displayed in this list. The default operation *types* are `Create`, `Get`, `Set`, and `Service`. The default operation *names* are `Create`, `Get`, `Set`, `Service`, and `GetList`. Each operation is defined by its name and type. When you add an operation, the Name field is automatically filled in. You can add multiple operations of the same type.

   For `Set` operations, there are two additional parameters. You can choose to set all the fields on the form, or you can choose a partial or full composite option. See "Set operations with line items" on page 344 for situations in which this is applicable.

11 Customize operations as required.

   To create an operation

   Default mappings are set when you create a new operation.

   a Right-click the Port tab and choose Add Operation.

   b Select an operation type from the list.

   c In the Name field, enter a name for the new operation.

   To copy an operation

   Copying an operation retains the existing mapping information.

   a Right-click the tab corresponding to the operation you want to copy.

   b Click Add Operation, then choose Copy of Selected Operation. The new operation appears under a new tab in the WSDL Operations list.

   c In the Name field, enter a name for the new operation.

   To delete an operation

   a In the WSDL Operations list, right-click the operation.

   b Click Remove.

   To change the name of an operation

   a Locate the operation in the WSDL Operations list, and expand the corresponding tab.

   b In the Name field, enter the new name.

12 (optional) In the Qualification field, enter a qualification.

When you select a `Set` or `Get` operation, the Qualification field is enabled. You cannot use an attachment field as a field reference in a qualification.

- For the `Set` operation, you can enter a query that enables the web service to find the request ID of the fields involved if the request ID is not known.
- For the `Get` and `GetList` operations, you can enter a query that identifies the field whose value you want to pass to the web service as the output parameter.
- If you select a `GetList` operation, the Start Record and Max Limit fields are populated with the appropriate paths. See "Setting the start record and the maximum limit" on page 337.

13 Map your parameters.

See "Mapping to simple and complex documents" on page 340.

14 In the Properties tab, set Permissions, Change History, and Help Text.

The permissions are the same (Public) whether the web service is visible or hidden.

IMPORTANT

If you publish your web service over an internet or intranet for general use, set the permissions to Public.

15 Click File > Save to save your web service.

The name of the web service should be descriptive and indicative of the web service's function. The name should not be the same as any existing active link guide, filter guide, packing list, or application.

16 Click the WSDL Publishing Location tab.

A sample URL for your WSDL file is displayed in the "Specify mid-tier's WSDL handler URL" field (see Figure 6-6 on page 75).

17 Adjust the URL for your configuration:

a Replace <*midtier_server*> with the name of the web server where the mid tier is running.

b Add `/public` or `/protected` after "WSDL," depending on the permissions of the web service.

c Replace "Untitled Web Service" with the name of the web service.

For example, if the web service has public permissions, use this format:

```
http://midtierServer/arsys/WSDL/public/ARServer/webServiceName
```

If the web service does not have public permissions, use this format:

```
http://midtierServer/arsys/WSDL/protected/ARServer/
webServiceName
```

18 Click View.

Your WSDL file is displayed in the tab. (See Figure 6-6 on page 75.)

19 After all adjustments are made, save the web service.

When you save the web service in BMC Remedy Developer Studio, the program creates a WSDL file that describes the web service. This file contains all the details necessary to interact with the service, including message formats, transport protocols, and end point location.

The WSDL file can be accessed with a URL using one of the following syntaxes.

- If the web service has public permissions, use this syntax:

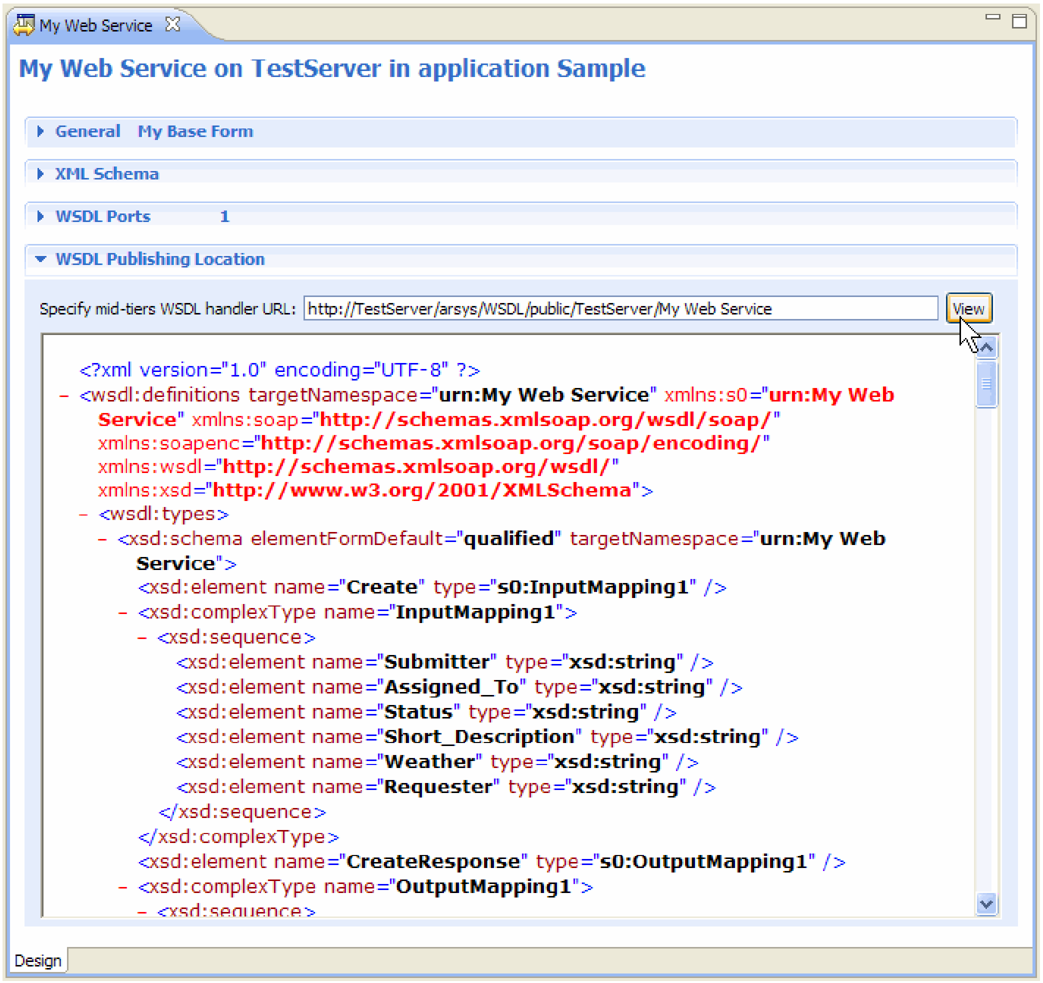
  `http://`*`midtierServer`*`/arsys/WSDL/public/`*`ARServer`*`/`*`webServiceName`*

- If the web service does not have public permissions, for example, if it used only inside a private network, use this syntax:

  `http://`*`midtierServer`*`/arsys/WSDL/protected/`*`ARServer`*`/`
  `webServiceName`

After you save your web service, it is ready to use. To verify that your web service is ready, enter the WSDL URL into a browser address field.

Figure 6-8: WSDL file displayed in WSDL panel of web services editor



Specify mid-tiers WSDL handler URL: http://TestServer/arsys/WSDL/public/TestServer/My Web Service

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="urn:My Web Service" xmlns:s0="urn:My Web
    Service" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <xsd:schema elementFormDefault="qualified" targetNamespace="urn:My Web
        Service">
        <xsd:element name="Create" type="s0:InputMapping1" />
      - <xsd:complexType name="InputMapping1">
        - <xsd:sequence>
            <xsd:element name="Submitter" type="xsd:string" />
            <xsd:element name="Assigned_To" type="xsd:string" />
            <xsd:element name="Status" type="xsd:string" />
            <xsd:element name="Short_Description" type="xsd:string" />
            <xsd:element name="Weather" type="xsd:string" />
            <xsd:element name="Requester" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="CreateResponse" type="s0:OutputMapping1" />
      - <xsd:complexType name="OutputMapping1">
        - <xsd:sequence>
```

Viewing a list of web services

You can view a list of available web services stored in AR System by entering the following URL in a browser:

`http://midtierServer/arsys/WSDL/protected/list`

Registering a web service

When you publish a web service, you need to provide a way for the users of the web service to find it. One way to do this, such as with a private web service used only on the company network, is to provide the URL of the WSDL directly to the users or build it into the client application. However, for many applications it is preferable to allow the client application to discover the web service at run time. To do this, the developer can register the service with a web services registry.

A web services registry is a database that stores descriptions of available web services and their end point URLs, using a standard description language. AR System developers can register, modify, deregister, and query registry entries for AR System web services and any other registered web service in the BMC Atrium Web Services Registry. Installation programs for web service applications can use AR System to register a web service at install time. At run time, AR System applications can use the BMC Atrium Web Services Registry to locate web services and obtain web service definitions.

The BMC Atrium Web Services Registry is installed with BMC Atrium Core, and is compliant with the UDDI standard. AR System integrates with this registry through a set of forms, workflow, and plug-ins.

This section describes how to use the AR System integration to this registry. For information about the BMC Atrium Web Services Registry itself, including a description of its metadata elements, see the *BMC Atrium CMDB 7.6.04 Developer's Reference Guide*.

NOTE

Using a web service registry is not required for AR System web services. If a registry is not available, the web service developer can provide the end point directly in the application. If the end point changes after the application is installed, use the utility `arwsendpoint.jar` to modify the address.

Web Services Registry prerequisites and configuration

This section describes the components of BMC Atrium Core and AR System that must be installed to support AR System use of the BMC Atrium Web Services Registry. It also describes the necessary AR System configuration settings.

To install and configure the BMC Atrium Core components described in this section, see the *BMC Atrium Core 7.5.00 Patch 001 Installation Guide.*

Prerequisites

To use the BMC Atrium Web Services Registry, the following BMC Atrium Core and AR System components must be installed and running:

- BMC Remedy AR System server and Java plug-in server
- BMC Remedy Mid Tier
- BMC Atrium Web Services Infrastructure (installed with BMC Atrium Core)
- BMC Atrium Web Services Registry (installed with BMC Atrium Core)

Configuring the Web Services Registry integration

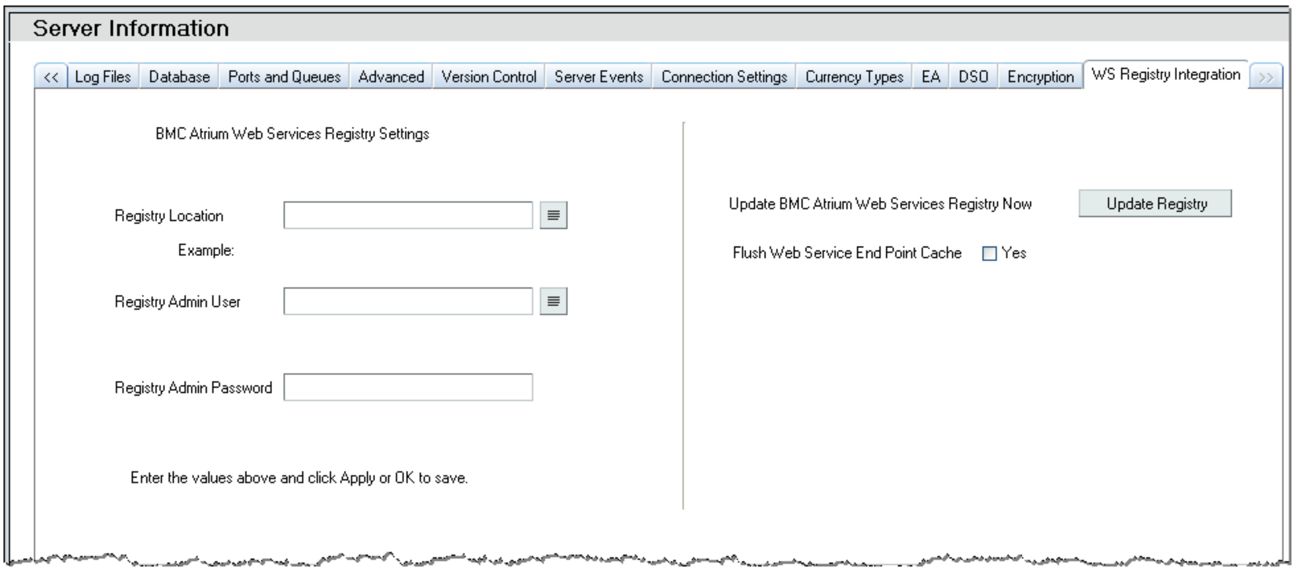
To activate the connection to the BMC Atrium Web Services Registry, use the WS Registry Integration tab of the AR System Administration: Server Information form.

▶ To configure the connection to the BMC Atrium Web Services Registry

1 In a browser or BMC Remedy User, open the AR System Administration Console, and click System > General > Server Information.

2 In the The AR System Administration: Server Information form, click the WS Registry Integration tab.

Figure 6-9: Server Information form—WS Registry Integration tab



3 In the BMC Atrium Web Services Registry Settings area, enter the options:

- **Registry Location**—The URL of the BMC Atrium Web Services Registry.
- **Registry Admin User**—The user name of the administrator for the web services registry.
- **Registry Admin Password**—The password of the web services registry administrator.

4 Click Apply.

NOTE

If the Registry Location is changed after AR System has registered web services and categories in the registry, AR System updates the new registry location with the existing web services and categories, but does not delete the web services and categories from the old location.

Plug-in server configuration file entries

Two registry plug-ins communicate with the BMC Atrium Web Services Registry API. A filter API plug-in (`ARSYS.ARF.REGISTRY`) handles all updates to the registry as a result of workflow on the AR System Web Services Registry, Category, and Registry Pending Delete forms. An ARDBC plug-in (`ARSYS.ARDBC.REGISTRY`) handles queries from the AR System Web Services Registry Query form.

The AR System installation program installs these plug-ins and creates the necessary configuration file entries. The AR System configuration file (`ar.conf` or `ar.cfg`) entries are:

- `Server-Plugin-Alias: ARSYS.ARF.REGISTRY ARSYS.ARF.REGISTRY pluginServerHostName:pluginServerPort`
- `Server-Plugin-Alias: ARSYS.ARDBC.REGISTRY ARSYS.ARDBC.REGISTRY`

In the Java plug-in server configuration file (`pluginsvr_config.xml`), the installation program creates XML `<plugin>` definitions for `ARSYS.ARF.REGISTRY` and `ARSYS.ARDBC.REGISTRY`. See the `pluginsvr_config.xml` file for the complete XML definitions.

For information about the BMC Atrium Web Services Registry API, see the *BMC Atrium CMDB 7.6.04 Developer's Reference Guide.*

Managing web services registry entries

AR System provides the following forms that support the BMC Atrium Web Services Registry integration:

Form name	Purpose
AR System Web Services Registry	The AR System interface to the BMC Atrium Web Services Registry. Creating a request in this form and marking it for publication registers a web service.
AR System Web Services Category	A supporting form that stores web service categories to help identify the web service. Each entry in the Category form is associated with one entry in the Registry form. An entry in the Registry form can be associated with one or more entries in the Category form.
AR System Web Services Registry Pending Delete	A supporting form that temporarily stores entries marked for deregistration.
AR System Web Services Registry Query	A vendor form that works with the ARDBC registry plug-in to query the registry at design time.

The following sections describe how to work with these forms:

- "Registering, modifying, and deregistering web services" on page 83—Register and deregister web services, and use categories to further define registry entries.
- "Exporting and importing data in the Registry and Category forms" on page 88—Best practices for importing and exporting data from these forms, for example, when distributing an application.
- "Querying the registry" on page 91—Use the Registry Query form to find existing registry entries by business name, web service name, application name, or category.

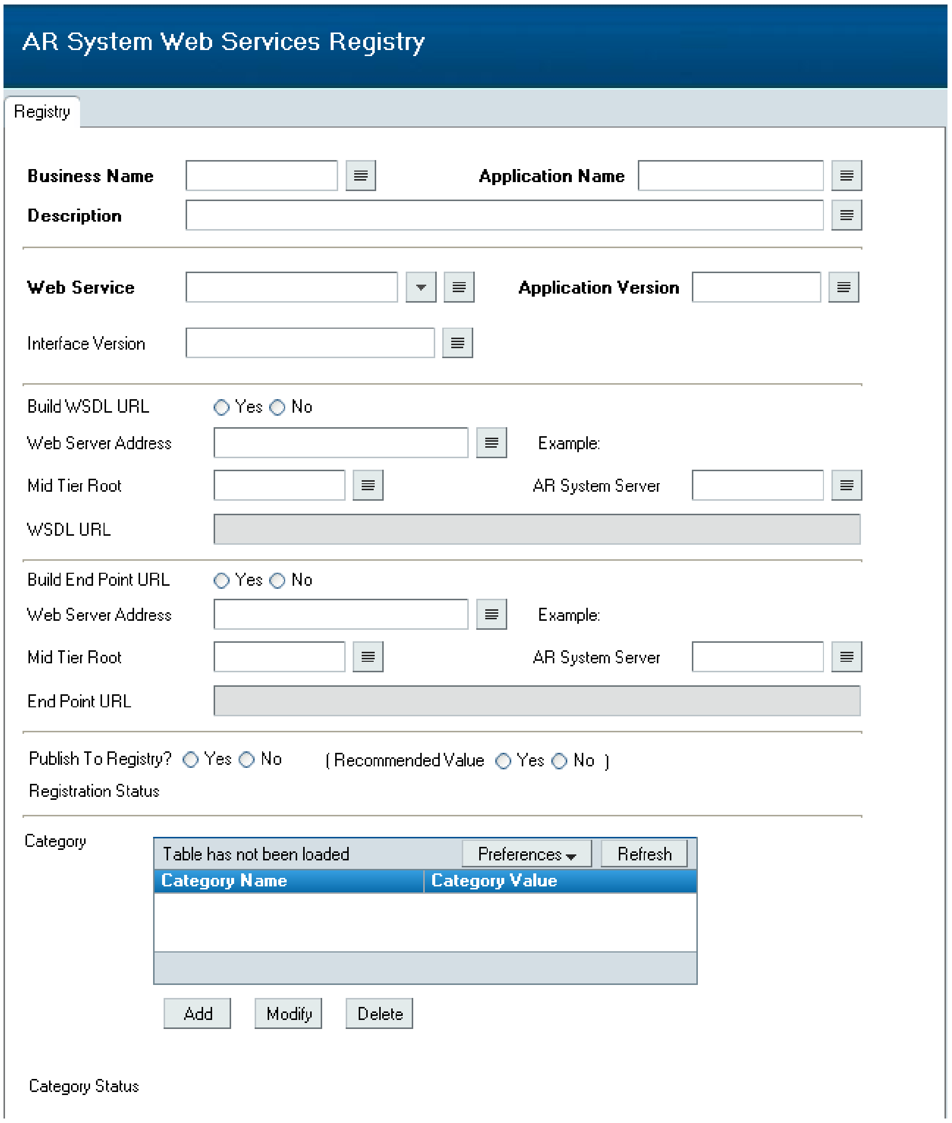
Registering, modifying, and deregistering web services

The procedures in this section describe how to use the AR System Web Services Registry form (Figure 6-10) and the AR System Web Services Category form to register, modify, and deregister web services. To use the procedures in this section, the web service must already be defined. See "Creating a web service" on page 72.

▶ To register a web service

1 Open the AR System Web Services Registry form in New mode in a web browser or in BMC Remedy User.

Figure 6-10: AR System Web Services Registry form



AR System Web Services Registry

Registry

Business Name		**Application Name**	
Description			
Web Service		**Application Version**	
Interface Version			
Build WSDL URL	◯ Yes ◯ No		
Web Server Address		Example:	
Mid Tier Root		AR System Server	
WSDL URL			
Build End Point URL	◯ Yes ◯ No		
Web Server Address		Example:	
Mid Tier Root		AR System Server	
End Point URL			
Publish To Registry?	◯ Yes ◯ No	(Recommended Value ◯ Yes ◯ No)	
Registration Status			

Category

Table has not been loaded

Category Name	Category Value

Category Status

2 Supply the required information in the following fields:

- **Business Name**—Use a consistent name to identify your organization.
- **Application Name**—Use a name that identifies the application that provides the web service.

  The web service is registered in the BMC Atrium Web Services Registry in the `<businessEntity>` named *businessName:applicationName*. The values are taken from the Business Name and Application Name fields in the form. The form workflow creates the `<businessEntity>` entry in the registry if necessary.
- **Description**—Your description of the web service.
- **Web Service**—The web service to register. Either select an AR System web service from the drop-down list or type the name of another web service.

- **Application Version**—The version of the application for which the web service exists.
- **Interface Version**—The version of the web service client API that defines the methods used in the web service.
- **WSDL URL**—The URL used to retrieve the WSDL for the web service.

  If this registry entry is for an AR System web service, Developer Studio can build the WSDL URL. Select Yes for Build WSDL URL and type the components of the URL in the fields provided:

  - **Web Server Address**—The URL of the web server that provides the WSDL. Include the port number if required.
  - **Mid Tier Root**—The root name of the AR System mid tier.
  - **AR Server**—The name of the AR System server that hosts the web service.

  If this registry entry is for an external web service, you must type the WSDL URL. Select No for Build WSDL URL, and then type the URL in the WSDL URL field.
- **End Point URL**—The URL used to call the web service.

  If this registry entry is for an AR System web service, Developer Studio can build the End Point URL. Select Yes for Build End Point URL and type the components of the URL in the fields provided.

  If this registry entry is for an external web service, you must type the End Point URL. Select No for Build End Point URL and then type the URL in the End Point URL field.
- **Publish To Registry?**—Controls whether the information in the request is sent to the BMC Atrium Web Services Registry.

  - Select Yes to have the form workflow make the entry in the registry consistent with the information in the request.
  - Select No to be able to create the request without updating the registry.
- **Recommended Value**—Specifies whether the default setting for Publish to Registry was Yes or No when the application was developed. An application author sets this value to recommend whether this web service should be registered, and the customer should not change this value.
- **Category**—A table field in the Registry form that lists any categories assigned to the web service definition. This field is supported by the AR System Web Services Category form, which opens when you take any action in the table field. The Category form is shown in Figure 6-11.

  - To add a new category to the registry entry, click Add in the Category table field of the Registry form.

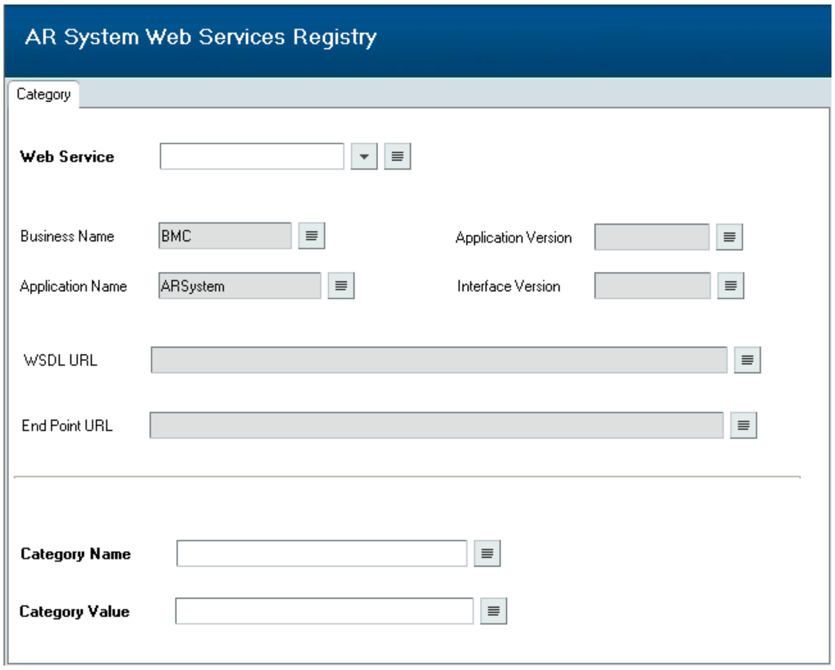
    The Category form opens and is populated with information from the current Registry form. Enter the appropriate values in the Category Name and Category Value fields. You can use any format in the Category Name and Category Value fields, except semicolons.

To select from a list of existing entries, click the field menu button on the Category Name field.

IMPORTANT

Do not use a semicolon (;) as part of a category name or category value. The semicolon is used to separate category name and value pairs when the registry is queried.

Figure 6-11: The AR System Web Services Category form



Business Name: BMC

Application Name: ARSystem

- To modify a category already associated with the registry entry, select the category in the table and click Modify. Then make the appropriate changes in the Category form.

TIP

If you do not see a recently added category in the field menu for the Category Name field, close and reopen the AR System Web Services Category form. This forces the list to refresh.

3 Save the Category form entry, if any, and the Registry form entry.

When you save the entries, workflow sends the changes to the registry plug-in, which in turn sends the updates to the registry. After the registry updates have been made, workflow updates the Registration Status field to "Registered."

▶ To remove an entry for a web service from the registry

1 Open the AR System Web Services Registry form in Search mode in a web browser or in BMC Remedy User.

2 Select the AR System web service from the Web Service list or type search values in other fields.

3 Click Search.

4 Select the correct request from the search results.

5 For Publish To Registry?, select No.

6 Save the request.

The form workflow removes the entry from the BMC Atrium Web Services Registry. After the entry is removed, workflow updates the Registration Status field to "Not Registered."

You can also remove an entry from the registry by deleting the corresponding request from the AR System Web Services Registry form.

NOTE

If an entry in the AR System Web Services Registry form is de-registered, the related entries in the AR System Web Services Category form remain. If an entry in the Registry form is re-registered, it is created in the Registry along with the corresponding entries in the Category form. If an entry in the Registry form is deleted, then the corresponding entries in the Category form are also deleted.

▶ To modify an existing registry entry for a web service

1 Open the AR System Web Services Registry form in Search mode and find the correct request in the form.

2 Change the appropriate fields in the request.

3 Save the request.

Synchronizing the Registry and Category forms with the registry

By default, workflow on the Registry and Category forms attempts to update the registry when you save entries in those forms and the field Publish to Registry is set to "Yes." This is usually sufficient to keep the registry in sync with the entries in the Registry form. However, the two can get out of sync if the registry is not available at the time an update is attempted, or if changes are made directly to existing AR System entries in the registry. In this case, the registry can be re-synchronized with the entries in the Registry form by two methods: workflow installed with AR System (automatic) and a forced update (manual).

Automatic registry synchronization

The "AR System Web Services Registry - check and update" escalation runs daily at 2 a.m. For each entry in the Registry form, the escalation compares the values in the Publish to Registry? and Registration Status fields. It updates the registry for any entry where Publish to Registry? and Registration Status do not match. Conditions that cause an update include:

- Publish to Registry? is set to “Yes”, but Registration Status is not set to “Registered.”
- Publish to Registry is set to “No”, but Registration Status is set to “Registered.”

This update includes all Category form entries associated with the Registry form entries that are updated. Administrators can modify this escalation to change the frequency and time if necessary.

Forcing an update to the registry

You can also use the Update Registry button on the WS Registry tab of the AR System Administration: Server Information form to force a registry update. In this case, AR System pushes to the registry all entries in the Registry form for which the Publish to Registry? field is set to “Yes”, regardless of the registration status. The associated entries in the Category form are also all pushed to the registry.

NOTE

This procedure does not move entries made directly in the registry into AR System. To add a new registry entry to AR System, you must create the entry in the Registry form.

▶ To force an update to the registry

1 In a browser or BMC Remedy User, open the AR System Administration Console, and select System > General > Server Information.

2 On the WS Registry Integration tab of the AR System Administration: Server Information form, click Update Registry.

Workflow attempts to update the BMC Atrium Web Services Registry to reflect the requests in the AR System Web Services Registry form.

Exporting and importing data in the Registry and Category forms

If you deploy a web service to another AR System server and you want to also deploy the related registry entries, you must export the related Registry and Category form entries along with the application that contains the web service.

Use the following guidelines to optimize the performance of registry updates when importing to the destination server:

- When the Registry form entry does not already exist on the destination server, import the Category form entries first, and then import the Registry form entries. In this case the registry update is triggered by the Registry form entry and all the registry updates are made in one call by the plug-in server. (In other words, if an entry is imported to the Category form and the corresponding Registry form entry does not yet exist, the registry server is not updated.)

- If you import Category form entries and the corresponding Registry form entry already exists, then the plug-in makes a separate call to update each Category form entry. For this reason, modifying a large number of existing Registry and Category form entries by importing them has a potential performance impact

Consuming a web service

Consuming a web service is using an external private or public web service to obtain information for use in an AR System application. This section describes how to create and configure a filter to obtain the web service information.

Creating a Set Fields web service filter action

To use an external web service, you create a Set Fields filter action with WEB SERVICE as the data source to enter data from the web service into the base form. You can then view the form in an AR System client. The Set Fields filter action invokes the web service using its WSDL.

Make sure that other filters are not acting on the same form that might skew the data from the external web service or prevent the data from appearing.

▶ To create a Set Fields web service filter action

1 In BMC Remedy Developer Studio, create a form to use as your base form for the external web service.

  Before you create fields to hold the web service data, review the XML element types in the WSDL file of your external web service. You can map only fields and elements of the same type. See “Data types” on page 360.

2 Choose File > New > Filter.

3 Select the server on which you want to create the filter, and click Finish.

4 Right-click the Associated Forms panel, then choose Expand All Panels.

5 In the Associated Forms panel, click the Add button.

6 In the Form Selector dialog box, select the base form and click OK.

  The data is set to this form. To push the data to other forms, create workflow.

  TIP
  Use the filtering options and the Locate field to navigate through the list of forms.

7 In the State list, choose Enabled.

  You might want to disable the filter during development, or when you detect a problem.

8 In the Execution Order field, enter the execution order for the filter.

The value you enter in this field determines the order in which the filter executes relative to other filters with the same triggering condition. Numbers between 0 and 1000 are valid values; lower numbers are processed first. Some filter actions might be queued and performed at a later time.

9 Select a check box corresponding to the operation that activates the filter.

   If you select multiple options, the filter executes when any of your selected operations runs.

10 To refine the selection criteria, enter a qualification statement in the Run If Qualification panel.

   When the qualification is met, If Actions are executed. When the qualification is not met, Else Actions are executed.

   TIP

   You can type or paste the qualification, or you can use the Expression Editor to build it. To open the Expression Editor, click the ellipsis button [...] adjacent to the field.

11 In the Error Handler panel, verify that error handling is disabled.

12 Right-click the If Actions panel or the Else Actions panel, choose Add Action, and select Set Fields.

13 From the Data Source list, choose WEB SERVICE.

14 From the Server Name list, select the server on which to store the web service mappings as a server object.

15 In the WSDL File field, enter the URL for the WSDL file of your external web service.

   a To select a WSDL file located on your local hard drive or LAN, click the ellipsis button and navigate to the WSDL file.

   b To search for the WSDL of a web service that is registered in the BMC Atrium Web Service Registry, use the AR System Web Services Registry Query form. See "Querying the registry" on page 91. Cut and paste the resulting WSDL URL into the WSDL File field.

16 When Developer Studio parses the WSDL, the End Point field appears and is populated with the end point stored in the WSDL. You can use the existing end point or modify it as follows:

   - Directly enter a different end point URL.
   - Enter an expression to cause AR System to query the registry at runtime to obtain the end point URL. See "Obtaining an End Point URL at run time" on page 93.

17 Click Reload.

   BMC Remedy Developer Studio parses the WSDL file, identifies viable operations, and lists them in the Operation list.

18 If prompted, enter the user name and password required by the remote web server for basic authentication.

19 From the Port list, select the web service Port from which to choose the operation.

20 From the Operation list, select the operation for the web service to perform.

21 Select an authentication method from the Authentication list. (See "Authentication information for consuming a web service" on page 97.)

22 Create input and output mappings. For mapping procedures, see "Mapping to simple and complex documents" on page 340.

23 Click File > Save to save the filter.

To view the web service data in an AR System client, open the base form and other forms to which the data is pushed.

Querying the registry

To find a WSDL or end point URL for a web service in the registry when configuring a Set Fields filter action to consume a web service, you can search the registry using the AR System Web Services Registry Query form in BMC Remedy User or a browser.

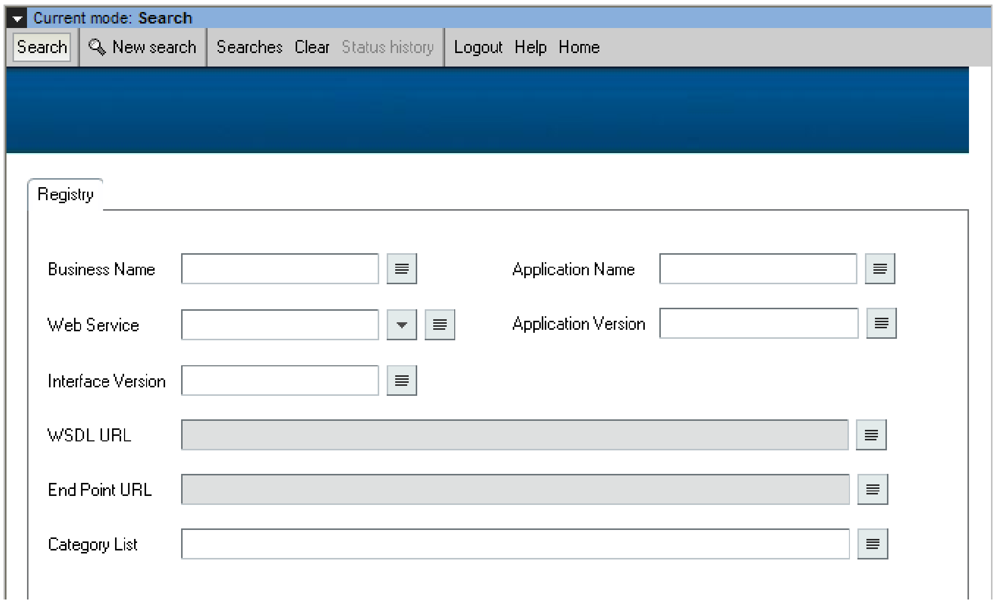
To cause AR System to query the registry for an end point URL at run time, you can use a special syntax in the End Point field of the Set Fields filter action.

This section describes both methods.

Using the Registry Query form at design time

To search the registry for a registered web service and obtain the WSDL or End Point URL for use in a Set Fields filter action, follow the procedure in this section using Registry Query form, shown in Figure 6-10.

Figure 6-12: The AR System Web Services Registry Query form



▶ To query the registry

1 In BMC Remedy User or a browser, open the AR System Web Services Registry Query form in Search mode.

2 Enter the known information about the web service in the form and click Search.

- You can use Java regular expression wildcards in the fields Business Name, Application Name, Web Service, and Version. For example, enter `BM*` to match “BMC Software, Inc.” or `BM.` to match “BMC”.
- You can search on all fields in the form except the WSDL URL and End Point URL fields. These fields are populated by the search results.
- Search values are hierarchical. You can search on either the Business Name or the Application Name alone, but to search the remaining fields, you must also enter values in each of the preceding fields. For example, to search on the Web Service field, you must also enter values in the Business Name and Application Name fields. The field hierarchy is Business Name >Application Name >Web Service >Interface Version > Category List.
- To use more than one category in the search, use a semicolon-separated list in the Category List field, for example, `name1=value1; name2=value2`. In the search results, the Category list is also populated using this syntax.

3 When the search results are returned, open the correct entry, and then cut and paste the returned WSDL into the Set Fields action in the Web Services filter.

AR System parses the WSDL and the web service end point appears in End Point field.

Obtaining an End Point URL at run time

To cause AR System to search the registry for the end point URL at run time, you enter a search phrase using a keyword, a registry key, and an optional list of categories in the End Point field of the Set Fields filter action. The registry key must provide enough information to obtain an unambiguous result. The key can include a list of categories associated with the web service. At run time, AR System builds the key using the values provided and retrieves the correct end point URL from the Registry.

To build the registry key, you use the keyword `REG::`, followed by name-value pairs separated by semicolons, in the following format:

REG::BN=*businessName*;AN=*appName*;W=*webService*;WV=*webServiceVersion*

For example:

```
REG::BN=BMC;AN=ARSystem;W=HelpDesk;WV=5.0
```

The element `REG`, followed by two colons (::), acts as a keyword to direct AR System to search the registry for the correct entry

The registry key consists of these elements:

`BN`—The Business Name, a required field.

`AN`—The Application Name, a required field.

`W`—The Web Service name, a required field.

`WV`—The Web Service Version, an optional field. If there is no web service version, this field is not required.

To further narrow the search, you can add an optional list of categories following the registry key. The registry key must precede the category list.

The category list is also constructed of name value pairs separated by semicolons, in the format:

```
CN1=CV1; CN2=CV2
```

In this format, `CN` is replaced by the Category Name and `CV` by the Category Value, for example, `Location=Paris; Owner=Marie`.

The category list is separated from the registry key by another set of two colons (::). For example:

```
REG::BN=BMC;AN=ARSystem;W=HelpDesk;WV=5.0::Location=Paris
```

You can enter the category list directly if you know the category names and values. Alternatively, you can enter field names in which the values will be supplied. For example:

`REG::BN=BMC;AN=ARSystem;W=HelpDesk;WV=5.0::$8$` or `$8$=$536870913$`.

If you use field references, test to verify that they are correct. The AR System server returns an error if the value is incorrect, which will cause the filter action to fail.

WSDL limitations for consumption

Most WSDL files are accepted during consumption, but some files can cause problems:

- SOAP-encoded arrays and SOAP-encoded structures are not supported. This means that RPC-encoded and document-encoded web services with complex input or output parameters do not work. For example, Amazon Web Services API and Google Web Services API fall into this category.
- All operations should be of one kind—that is, all rpc encoded or all doc/literal.
- Only SOAP operations are considered. MIME and HTTP operations are ignored.
- Overloaded operations are not allowed.
- Both input and output should be present; one-way messaging is not allowed.
- A WSDL file cannot have both a `<wsdl:include>` and a `<wsdl:types>` element.

TIP

As a workaround, use `<xsd:include>` inside `<wsdl:types>`. There is no restriction on the number of `<xsd:include>` elements that you can use.

- A WSDL file cannot have more than one `<wsdl:include>` element.
- AR System supports Message Parts that point to XML Elements or XML Simple Types only.

Managing web service performance issues

This section describes configuration changes that might be required when using web services with AR System.

Server threads

If an AR System server calls itself through a web service, twice the number of fast and list threads are used. Therefore, the minimum number of fast and list threads should be more than two. The number of server threads should be two times the expected number of users that will use the web-services feature if they are consuming on the same server.

NOTE

It is easy to get into a deadlock situation by running out of threads, because each web service call consumes two threads.

To consume a web service created on the same AR System server, you must increase the number of threads. To do so, open the AR System Administration: Server Information form, select the Ports and Queues tab, and adjust the number of threads.

Threads in the Plug-in

By default, the web service plug-in uses only one thread. If you use multiple web services, either external or within the AR System server, configure the plug-in with multiple threads. To specify the number of plug-in server threads for the filter API (which is used in `webservices.dll`), use the `Plugin-Filter-API-Threads:` *`minThreads maxThreads`* entry in the `ar.cfg` file. Configure the number of threads according to the load; initially, set the minimum number of threads to five. If the plug-in server does not respond in time, increase the minimum threads.

Time-out

If an external web service is too slow, AR System times out in 40 seconds, by default. To set the time-out to 20 seconds, set `Filter-API-Timeout:20`.

Log File

Set `Plugin-Log-Level:100` to log to a file specified in the Log Files tab of the AR System Administration: Server Information form.

SOAP headers and authentication

This section describes how AR System authenticates requests that come in to a web service published by AR System, and how to configure authentication information in order to consume an AR System or other external web service. Authentication information consists of a user name and password, which are included in the SOAP packet, usually in the header.

WARNING

Authentication information in SOAP headers or other web services communication can be in plain text. To ensure the security of the authentication information in a SOAP header in this case, configure the web server to use https.

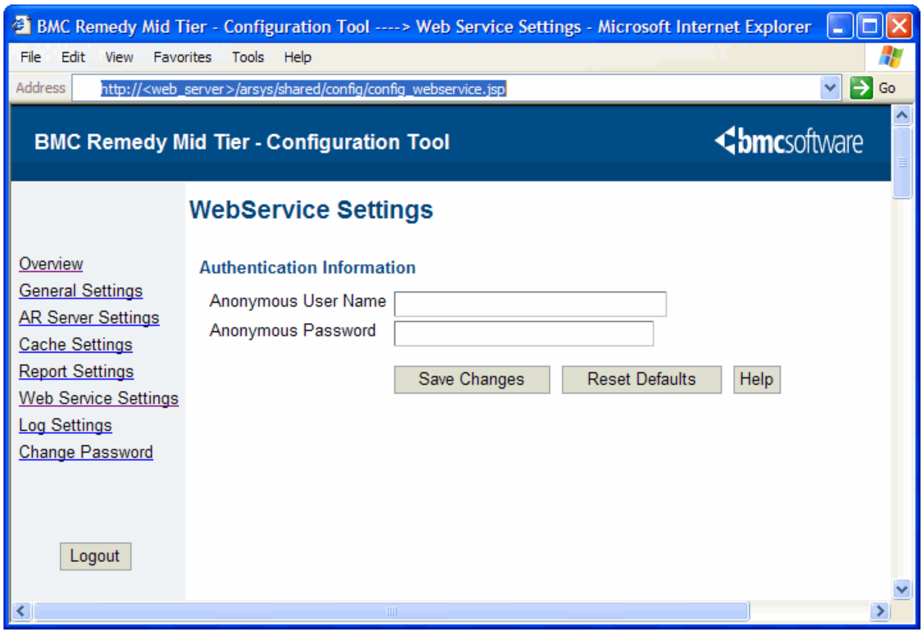
Authentication information for a published web service

When you publish a web service, the consuming application can specify which user is authorized to perform the web service operations. AR System checks whether the SOAP header contains the root XML data type `AuthenticationInfo`. This is an AR System XML data type, and consists of the child elements `userName`, `password`, `authentication`, `locale` and `timeZone`. The `timeZone` child element is optional. When the `AuthenticationInfo` data type is included in the SOAP header, it should look like the following example:

```
<AuthenticationInfo>
  <userName>Joe</userName>
  <password>ILoveDogs</password>
  <authentication>ARSystem</authentication>
  <locale>en_US</locale>
  <timeZone> </timeZone>
</AuthenticationInfo>
```

If the authentication information is not specified, AR System uses the Anonymous User Name and Anonymous Password defined in the-BMC Remedy Mid Tier Configuration Tool.

Figure 6-13: BMC Remedy Mid Tier Configuration Tool—Web service settings


BMC Remedy Mid Tier - Configuration Tool ----> Web Service Settings - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://<web_server>/arsys/shared/config/config_webservice.jsp

BMC Remedy Mid Tier - Configuration Tool

WebService Settings

Overview
General Settings
AR Server Settings
Cache Settings
Report Settings
Web Service Settings
Log Settings
Change Password

Authentication Information

Anonymous User Name
Anonymous Password

Save Changes | Reset Defaults | Help

Logout

You can use any name as the Anonymous User Name for development purposes, but for production, create a user name specific to the published web service.

Authentication information for consuming a web service

When you create workflow to consume an AR System or external web service that requires a SOAP header, BMC Remedy Developer Studio checks the SOAP header elements specified in the WSDL. If the WSDL specifies the SOAP header elements, Developer Studio automatically creates a `SOAPHeader` element under the `ROOT` element. To communicate with any web service that requires authentication, you must use the Input Mapping table to map the source of the user name and password in the web services workflow.

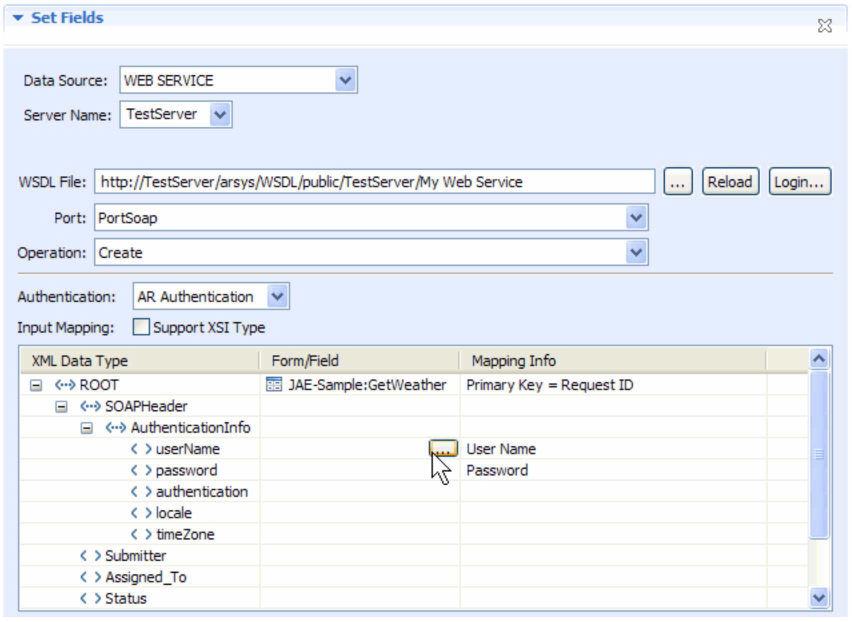
When you add a Set Fields action to a filter or escalation and choose WEB SERVICE as the data source, the Authentication field appears. Use this field to define the authentication type required for the web service. The options are:

- **None**—The web service does not require authentication.
- **Custom**—The external web service must be designed to recognize the fields that contain the user name and password. The SOAPHeader is not used to carry the user name and password. The password is sent unencrypted.
- **AR Authentication**—The web service is provided by an AR System server. This option uses the AR System `AuthenticationInfo` data type in the SOAP header. The password is sent unencrypted.
- **Username Token**—The web service is an external web service that supports WS-Security Username Token 1.0. (Web Services that use Username Token 1.1 can be used, if they support 1.0 as well.) The password can be sent either as plain text or as a digest as specified in the WS-Security specification.

Mapping the user name and password XML data types

To define the user name and password to be used when consuming the web service with a SOAP header, map the SOAP attributes for user name and password in the Input Mapping table of the Set Fields action. Figure 6-14 shows an example using the AR Authentication option.

Figure 6-14: Mapping authentication data types in the SOAP header—AR Authentication



Data Source: WEB SERVICE
Server Name: TestServer
WSDL File: http://TestServer/arsys/WSDL/public/TestServer/My Web Service
Port: PortSoap
Operation: Create
Authentication: AR Authentication
Input Mapping: ☐ Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	JAE-Sample:GetWeather	Primary Key = Request ID
SOAPHeader		
AuthenticationInfo		
userName		User Name
password		Password
authentication		
locale		
timeZone		
Submitter		
Assigned_To		
Status		

For each authentication type and based on the selected WSDL, the authentication elements in the XML Data Type column change. The mapping options in the Form / Field and Mapping Info columns also change depending on the Authentication type. In general, you can use any of these methods to map the user name and password values:

- Map the XML data types to character fields in the associated form.
- Provide static values (the `default` XML data type).
- Use the AR System user name and password of the user currently executing the filter.

You can also use a combination of these mappings. If the user name and password elements are mapped to fields, the AR System server first attempts to retrieve the values from those fields. If the field values are NULL, then the AR System server checks for the static value (in the `default` attribute) of `userName` and `password`, and uses it if present. If the mapped fields and the default values are NULL the current AR System user name and password of the user executing the filter action are used.

TIP
You can see the XML attributes and their values for any element in the XML Data Type column by hovering the mouse cursor over the element.

AR Authentication method

Use this method when the web service is hosted on an AR System server. In this case, Developer Studio automatically sets the XML attributes `arUsername: true` and `arPassword: true`. In the Input Mapping table, this is represented by the entries `User Name` and `Password` in the Mapping Info column. This allows AR System to use the current user's user name and password at run time.

▶ To enter static values for userName and password with the AR Authentication method

1 In the Form / Field column, click in the cell corresponding to the `userName` attribute.

2 Type the user name, and then press Enter.

3 Repeat steps 1 and 2 in the cell corresponding to the `password` attribute, typing the password.

   The password is obscured as you type.

▶ To map the userName and password attributes to fields in the form

1 In the Form / Field column, click in the cell corresponding to the `userName` attribute, and then click the ellipsis (...) button.

2 In the Field Selector dialog box, select the appropriate field, and then click OK. For example, map the Requester field to `userName`.

3 Repeat steps 1 and 2 for the `password` attribute.

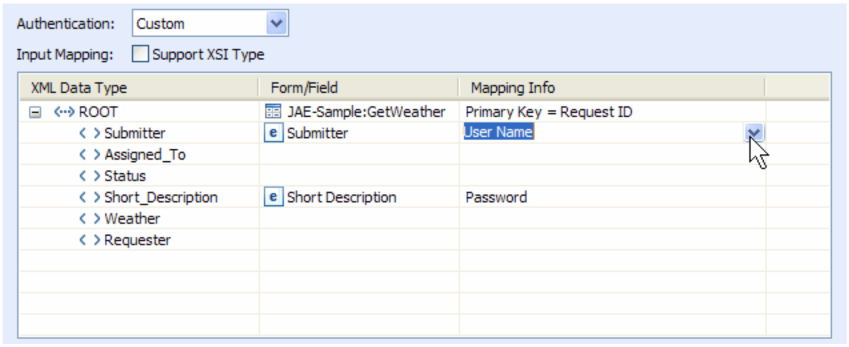
TIP

To configure the input mapping to have both a field mapping and a static value, enter the static value first. Then, click in the cell again *without deleting the static value*, click the ellipsis, and select the field to map. To verify that the static value has been assigned, hover the cursor over the appropriate field and check the value of the `default` attribute. If you delete the field mapping after doing this, the static value is also deleted.

Custom authentication method

Use this method for an external web service, not hosted by AR System, where the Username Token is not required. The consumed web service must be designed to recognize the fields that contain the user name and password. In this case, the SOAPHeader element is not used in the input mapping. Figure 6-15 shows the input mapping being configured for the Custom authentication method.

Figure 6-15: Input mapping for the Custom authentication method



Authentication: Custom

Input Mapping: ☐ Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	JAE-Sample:GetWeather	Primary Key = Request ID
Submitter	Submitter	User Name
Assigned_To		
Status		
Short_Description	Short Description	Password
Weather		
Requester		

With the custom method, you can enter a static value for the user name and password, map fields to the appropriate attributes, use the run time user's AR System user name and password, or any combination of the three. With the Custom method, when you use the Mapping Info column to identify which element the web service should use as the User Name and password, Developer Studio also sets the attributes `arUsername` and `arPassword` to `true`.

To map fields or assign static values for the user name and password, follow the procedures in "AR Authentication method" on page 98, selecting the appropriate XML data types that you want to use.

▶ To assign the User Name and Password flags to the appropriate elements

1 In the Mapping Info column, click in the appropriate cell for the XML data type you are using as the user name.

2 Click the drop-down menu icon that appears, and select User Name.

3 Press Enter.

If you click elsewhere on the table instead of pressing Enter, the attribute is not set correctly.

4 Repeat steps 1 through 3 for the XML data type you are using as the password.

Username Token authentication method

The WS-Security WS-Security Username Token 1.0 standard allows you to specify whether you send the password in digest or plain text format. Digest format is a hash, and the consuming web service must know the password in order to authenticate the password.

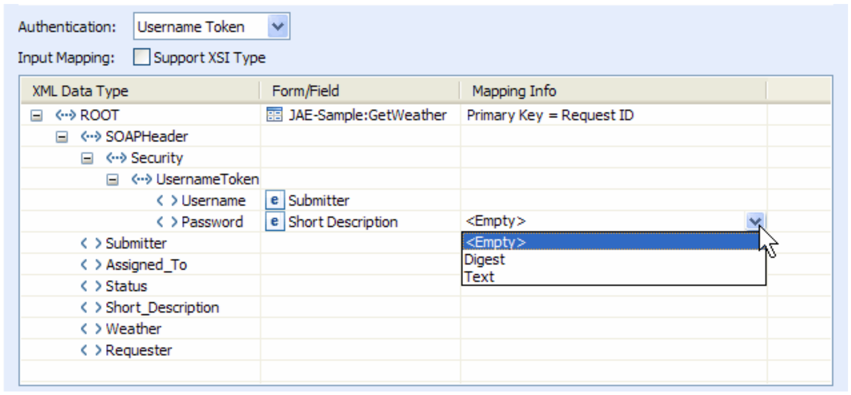
For documents describing the WS-Security Username Token 1.0 standard, see `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss`.

NOTE

For a web service that requires the Username Token, the selected WSDL might not indicate that a Username Token is required. In that case, the designer must know whether the web service requires the Username Token and create the workflow accordingly.

With the Username Token method, you can enter a static value for the user name and password, map fields to the appropriate attributes, use the run time AR System user name and password, or any combination of the three. With this method, Developer Studio automatically sets the XML attributes `arUsername: true` and `arPassword: true`, but this is not reflected in the Mapping Info column. For this method, the Mapping Info column defines whether to send the password in digest or plain text format.

Figure 6-16: Input mapping for the Username Token authentication method



Authentication: Username Token

Input Mapping: Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	JAE-Sample:GetWeather	Primary Key = Request ID
SOAPHeader		
Security		
UsernameToken		
Username	Submitter	
Password	Short Description	<Empty>
Submitter		
Assigned_To		
Status		
Short_Description		
Weather		
Requester		

To map fields or assign static values for the user name and password elements, follow the procedures in "AR Authentication method" on page 98.

▶ To assign digest or plain text format to the password element

1 Click in the Mapping Info cell for the `Password` element.

2 Click the drop-down menu that appears, and then select Digest or Text.

Supported schema constructs and AR System web service limitations

AR System supports most common schema constructs, but not all. AR System does not perform XML validation, so developers must check this list to make sure the application uses only supported schema constructs.

Supported XML schema constructs

The following XML schema constructs are supported in AR System:

Elements

- global **element** declaration
- local **element** declaration
- **nillable** for an **element**
- **minOccurs**=0 for an **element**
- **default** for an **element**
- an element that has a **ref** to another element

Attributes

- **attribute**
- **default** for attributes
- **attributeGroup**

Types

- named **simpleType**
- inlined **simpleType**
- named **complexType**
- inlined **complexType**

Derivation

- **complexType** derived from **complexContent** by extension
- **complexType** derived from **complexContent** by restriction
- **complexType** derived from **simpleContent** by extension
- **simpleType** derived from **simpleContent** by restriction
- **simpleType** derived from **simpleType** by restriction (The restriction is ignored)

Model group

- **sequence** directly under a **complexType** with **maxOccurs**=1 and **minOccurs**=1 (if **maxOccurs** and **minOccurs** are unspecified, they default to 1)
- **choice** directly under a **complexType** with **maxOccurs**=1 and **minOccurs**=1
- **all** directly under a **complexType** with **maxOccurs**=1 and **minOccurs**=1
- **sequence** directly inside a **sequence** with **maxOccurs**=1 and **minOccurs**=1
- **choice** directly inside a **choice** with **maxOccurs**=1 and **minOccurs**=1
- **choice** directly inside a **sequence** with **maxOccurs**=1 and **minOccurs**=1
- **group**
- **maxOccurs=unbounded** for an **element** which is the sole child of a **sequence**. In all other situations, maxOccurs must be 1 or not be specified. Instead of maxOccurs=unbounded, it could also be maxOccurs=20 (or some other number greater than 1) in which case any XML generated by AR System would have 20 or fewer elements. This element must be of a complexType.
- Namespaces
  - **targetNamespace**
  - chameleon namespaces

- Multiple documents
  - **include**
  - **import**
- Miscellaneous
  - **annotation**

The choice element

The choice element in the XML schema gives the flexibility of listing a set of elements. The XML instance document using this schema can specify only one of the elements mentioned for choice. But when you generate an output XML document from AR System forms, it is not obvious that you need to select only one of the choice elements. To resolve this, a choice node in the XML schema can be mapped to a character field. When an input XML document is received, the name of the element given for choice is stored in this character field. The value in this character field is used in generating the output XML document for choice.

In addition:

- Direct or indirect children of choice cannot be mapped to another form.
- Choice can have only elements. It cannot have immediate choice, sequence, group and so on.
- A choice can appear in a sequence or in a complex type.
- Recursion in choice is not allowed.
- "minOccurs" attribute of choice can be either 0 or 1. Any number greater than 1 or unbounded is not supported.
- "maxOccurs" attribute of choice must be 1. Any number greater than 1 or unbounded is not supported.
- Choice does not reset other items during a `Set` or `Create` operation. It sets only the item that is sent.

XML schema constructs not supported in AR System

The following XML schema constructs are *not* supported in AR System:

- Recursive definitions
- **list**
- **union**
- Type substitution
- An element of a simpleType having **maxOccurs**>1
- **sequence** with two elements having the same name
- **sequence** with **maxOccurs**>1 or **minOccurs**=0
- **choice** with **maxOccurs**>1 or **minOccurs**=0

- **all** with **maxOccurs**>1 or **minOccurs**=0
- **sequence** directly inside a **choice**
- Multiple **choices** under a **sequence**
- An element under a **choice** having **maxOccurs**>1
- Multiple elements under a sequence with one or more of them having **maxOccurs**>1 (if there is only one element that could have **maxOccurs**>1)
- **substitutionGroup**
- **redefine**
- **noNamespaceSchemaLocation**
- **any**
- **anyAttribute**
- **abstract**
- **mixed="true"**
- **ID**, **IDREF**, **NOTATION**, **normalizedString**, **NCName**, **ENTITY**, **token**, **language** (treated as strings, ignoring any restrictions)
- **Name** (not supported)
- **IDREFS**, **ENTITIES**, **NMTOKENS** (not supported)
- **key**, **unique**, **keyref** (ignored)

Chapter 7 Plug-ins

AR System plug-ins enable you to extend AR System *server* functionality to external data (data not contained in the AR System database). This section describes each type of AR System plug-in and provides some general information about plug-ins.

NOTE

To extend *client* functionality, you use the AR System C API or Java API. See Chapter 4, "AR System C API," Chapter 5, "Java API,", and the *C API Reference*.

The following topics are provided:

- About AR System plug-ins (page 106)
- Installing plug-in components (page 108)
- Creating C plug-ins (page 110)
- Creating Java plug-ins (page 112)
- Configuring the Java plug-in server (page 114)
- Configuring the AR System server (page 118)
- Running the plug-in server (page 120)
- Common plug-in C functions and Java methods (page 123)
- AREA plug-ins (page 124)
- ARDBC plug-ins (page 127)
- AR filter API plug-ins (page 132)

About AR System plug-ins

The AR System plug-in server allows integration between AR System and external programs or environments by managing the interaction between the plug-in code and the AR System server. All plug-ins are registered with the plug-in server, which runs them as needed and coordinates all interaction.

A plug-in is defined by using one of the plug-in APIs to write code to handle the integration with the external program. Plug-in API functions provide the main routine, threading control, and communication with the AR System server. The plug-in application that you write provides the logic for one or more callback routines, defined by the API, that perform operations against the external program or environment.

When a plug-in function is invoked, the AR System server makes a call to the plug-in server, requesting a specific plug-in to perform an operation with a set of parameters. The plug-in server passes the parameters to the appropriate callback routine in the external application and awaits the response. When the response is received, it is returned to AR System and processing continues.

AR System supports the following types of plug-ins that you create:

- **AR System External Authentication (AREA)**

  AR System uses AREA plug-ins to authenticate the identity of users. AREA plug-ins access network directory services or other authentication services to verify user login names and passwords. When you use an AREA plug-in, you do not have to maintain duplicate user authentication data in the AR System directories because the AR System server can access user identification information from external sources.

NOTE

If you have users with fixed licenses or users who use BMC Remedy applications, you must maintain user authentication data in AR System directories because users must exist in the User form for the license tracking feature and for the applications.

NOTE

See “AREA plug-ins” on page 124. In this release of AR System, a ready-to-use Atrium SSO plug-in is introduced for the purpose of single sign-on. This plug-in is used for integrating the AR System server and the Atrium SSO server and authenticates the user against the Atrium SSO server by calling the Atrium SSO APIs. For more information, see “Configuring Atrium SSO integration” on page 161.

- **AR System Database Connectivity (ARDBC)**

  ARDBC plug-ins enable AR System to access data stored in external sources. You can integrate ARDBC with the external data sources through their own APIs. ARDBC plug-ins, which you access through vendor forms, enable you to perform these tasks:

  - Search external data sources
  - Create, delete, modify, and set entries in external data sources
  - Populate search-style character menus from external data sources
  - Implement workflow that accesses external data sources

  See “ARDBC plug-ins” on page 127.

- **AR System Filter (AR Filter)**

  AR filter API plug-ins are used when server-side workflow objects, such as filters and escalations, reference filter API calls. AR filter API plug-ins offer an alternative method to send information requests to and from external servers. In previous versions of AR System, run processes performed external information requests. AR Filter uses fewer system resources than run processes use and enables the AR System server to return to its workflow faster.

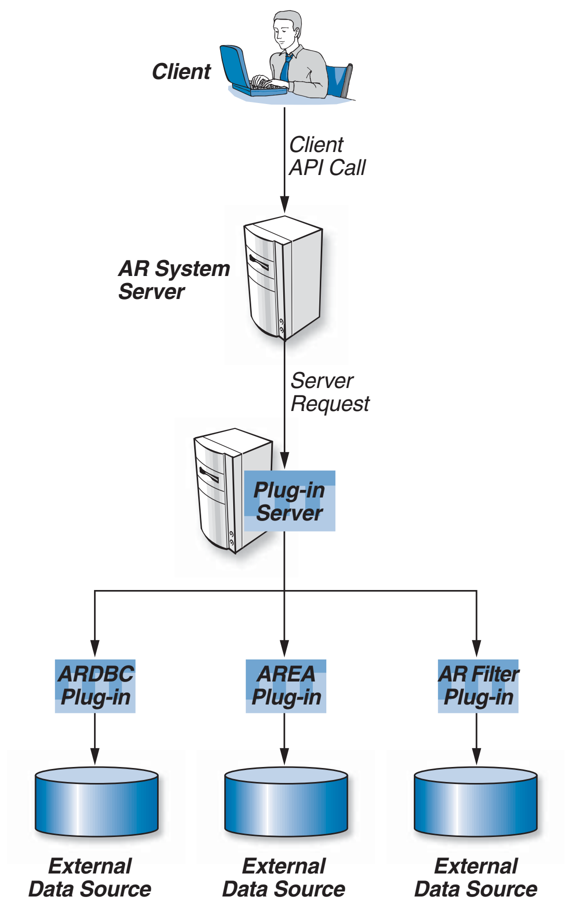
  See “AR filter API plug-ins” on page 132.

NOTE

AR System also offers two ready-to-use Lightweight Directory Access Protocol (LDAP) plug-ins that you access through BMC Remedy Developer Studio. See Chapter 8, “LDAP plug-ins.”

“Installed files” on page 109 shows how the AR System server calls the plug-in server that calls the plug-in.

Figure 7-1: AR System plug-in architecture



NOTE

The arrows in this figure identify the directions in which each program or process can initiate API function calls. Data can flow in any direction.

Installing plug-in components

Before you can create AR System plug-ins, you must install these components:

- **AR System plug-in servers**—Automatically installed with AR System.
- **API component of the AR System server**—Includes the plug-in APIs. You must select this option during the AR System installation. The API component includes:
  - The header files you use to compile C plug-ins and create the shared libraries. See the `arplugin.h` file for C plug-in definitions and declarations.
  - The files you need to write Java plug-ins.

For more information, see the *Installation Guide*.

NOTE

In addition to the AR System 7.6.04 Java plug-in server and its API, the C plug-in server, `arplugin`, and its API are installed.

Installed files

The plug-in server and Java plug-in API files are typically installed in the following directories:

- **UNIX**—`/usr/ar/`*`serverName`*`/pluginsvr`
- **Windows**—`C:\Program Files\AR System\`*`serverName`*`\pluginsvr`

Component	Description
`arapi75.jar`	Includes the AR System Java API, Java utilities, and AR System server communications. This file is called by Java plug-ins.
`arplugin.log`	Plug-in server log file. This file is generated when the Java plug-in server starts.
`arpluginjni75.dll`	(Windows) C plug-in server interface for C plug-ins.
`arpluginsvr75.jar`	Java plug-in server and plug-in interfaces. This file is called by the Java plug-in server.
`log4j_pluginsvr.xml`	Java plug-in server logging configuration file.
`log4j-1.2.14.jar`	Java logging utility.
`pluginsvr_config.xml`	Java plug-in server configuration file.
`pluginsvrstartup.bat`	Java plug-in server start-up file for Windows.
`pluginsvrstartup.sh`	Java plug-in server start-up file for UNIX.

The `arplugin.h` file is installed with the other C API include files in the `include` subdirectory.

Creating C plug-ins

You must create a separate plug-in for each of the three types of plug-ins. For example, you cannot create one plug-in that supports both AREA and ARDBC.

NOTE

On Windows platforms, plug-ins created for pre-7.0 servers must be recompiled with Microsoft Visual Studio .NET 2003 to be used successfully in the AR System 7.0 environment.

▶ To create a C plug-in

1 Write a C or C++ program that includes these elements:

- A reference to the `arplugin.h` file for plug-in definitions and declarations.
- Plug-in API calls for initialization, termination, object creation, and object deletion (see "Common plug-in C functions and Java methods" on page 123).
- One of these type-specific API calls:
  - AREA (see page 126)
  - ARDBC (see page 128)
  - AR Filter (see page 132)
- Code to implement the calls. Use sample Microsoft Developer Studio projects (Windows) or makefiles (UNIX), which you install with AR System, to build your program into your DLL or shared object library.

  For examples and templates for C plug-ins, see the `ardbc`, `area`, and `arfilterapi` subdirectories of the `Api` directory in your AR System server installation.

Figure 7-2 on page 124 shows the general structure of a plug-in program.

2 Compile and link your plug-in as follows:

- On Windows platforms, compile your plug-in using Microsoft Visual Studio .NET 2003.
- On Linux, you must use the `-malign-double` option when compiling plug-ins to make sure that your plug-in library is correctly aligned for the plug-in server. If you do not, the plug-in might produce unexpected results.
- If you compile and link your plug-in on HP-UX using aCC and enable exception handling, your plug-in will have dependencies on libraries that are not standard C/C++ libraries, for example `libCsup_v2` and `libstd_v2`. If your plug-in has dependencies on any libraries like these, you must explicitly link to them and make sure they are available at run time in the shared library path for the plug-in server to find them.

3 Put your plug-in DLL or shared library file in the directory that contains the AR System server and plug-in server executable files or any other directory listed in you `PATH` environment variable.

4 Add an entry for the plug-in to the plug-in server configuration file. See "Configuring the Java plug-in server" on page 114.

At run time, the plug-in server reads the configuration file and loads the specified plug-ins.

C plug-in conventions

When you create a C plug-in, use the following naming conventions and memory management practices.

Memory management

Do not free memory that the plug-in passes to your functions as arguments or that you return to the plug-in. Plug-ins can allocate and deallocate memory associated with the `object` argument for each call. The plug-in does not deallocate this memory.

Input and return values

In a C API program, developers specify input values for the functions and receive return values. In a plug-in program, the plug-in provides the input values to the plug-ins, and the plug-ins provide return values.

If the AR System server returns `AR_RETURN_WARN` or `AR_RETURN_OK` to the `arplugin` log file after a call is issued, the plug-in considers that call successful. The plug-in considers the call unsuccessful if the server returns `AR_RETURN_ERROR` or `AR_RETURN_FATAL`.

If you do not implement a call, the plug-in performs a default action. The default action might be to proceed or to return an error message.

Protection for global information

To ensure thread safety, you must protect any global information or resources that you access through plug-in API calls with appropriate mutual exclusion locks. Global information and resource protection applies to all plug-in calls except `ARPluginIdentify`, `ARPluginInitialization`, `ARPluginSetProperties`, and `ARPluginTermination`, which are always called by one thread at a time.

At run time, the plug-in server reads the configuration file and creates the plug-ins that the file specifies. After the plug-in server creates the plug-ins, they remain active until a system failure or until you modify the plug-in configuration information and restart the plug-in server. For information about restarting the plug-in server, see "Restarting the plug-in server using the Set Server Info command" on page 118.

Creating Java plug-ins

The Java plug-in API includes an interface and an abstract class for each plug-in type. Your Java plug-in program must implement one of the interfaces or extend one of the abstract classes.

Interface	Extends
`ARDBCPluggable`	`ARPluggable`
`AREAPluggable`	`ARPluggable`
`ARFilterAPIPluggable`	`ARPluggable`

Abstract class	Extends	Implements
`ARDBCPlugin`	`ARPlugin`	`ARDBCPluggable`
`AREAPlugin`	`ARPlugin`	`AREAPluggable`
`ARFilterAPIPlugin`	`ARPlugin`	`ARFilterAPIPluggable`

The interfaces and classes are described in detail in the Javadoc-generated online documentation in the `arpluginsdoc.jar` file. This file is located in the `javaplugins` subdirectory of the `Api` (or `api`) directory in your AR System server installation directory. To access the Java plug-in API documentation, unzip the contents of the file. (To unzip a JAR file, use a zip utility or the Java `jar` executable, which is in the `bin` directory of the Java JRE is installation. For example, `jar -xvf arpluginsdoc.jar`.) Then, navigate to the `javadoc` folder, and open the `index.html` file to see an overview of the entire AR System Java plug-in API documentation.

Classes, instances, and shared data

Two or more Java plug-in classes can be configured in a plug-in set or group as described in the comments in the `pluginsvr_config.xml` file. When the Java plug-in server starts, it loads each configured plug-in class or group in a separate class loader and any static initialization in the classes is executed. It also initializes an instance of each plug-in class listed in the `pluginsvr_config.xml` file for each worker thread in its worker thread pool. Each time the AR System server makes a connection to the Java plug-in server, a selector thread adds the request associated with the connection to a task queue. As soon as a worker thread is free, it processes the next request in the task queue.

Different instances of a class can share data in the static *class* variables. To be thread safe, however, the class implementation must protect this static data.

The class can use *instance* variables to store data that is not shared. Because each thread has a separate instance, this data is thread safe.

Writing a Java plug-in

▶ Follow these steps to write the program:

1. Make sure your programming environment is set up correctly. You need:
   - Java Development Kit (JDK) 5 or later. (Java SE 5 is also know as Java 1.5.0.)
   - AR System Java plug-in API files. (See "Installed files" on page 109 for installation and verification.)
   - AR System Java API files. (See "Installed files" on page 109 for installation and verification.)
2. Create a Java project in your IDE.
3. Include the `arpluginsvr75.jar` file in the AR System API library directory in the `CLASSPATH`. (For the directory location, see "Installed files" on page 109.)
4. Create a new class that will implement one of the interfaces listed in "Creating Java plug-ins" on page 112 or extend one of the abstract classes listed in that section.
5. Import the `com.bmc.arsys.pluginsvr.plugins` and `com.bmc.arsys.api.*` packages and other packages, such as `java.util.List`, into your program.
6. Implement the methods of the interface or abstract class you are using. See the Java plug-in API online documentation for details, and consider these tips:
   - When implementing the `init()` and `initialize()` methods, do not put the plug-in into a long loop to wait to connect to the AR System server or another server. Such loops prevent the Java plug-in server from finishing its initialization. To determine whether the plug-in has been instantiated inside the plug-in server, the plug-in server must receive either a return from `init()` or `initialize()`, or an exception. If the plug-in server learns that the method failed to instantiate the plug-in, it can still instantiate its other plug-ins and complete its initialization. The failed plug-in, however, will be unable to receive calls.
   - If a plug-in must perform a long process, such as establishing a database connection, before the plug-in can accept a call, create a separate thread for the process so that the `init()` and `initialize()` methods are not blocked. If the plug-in receives any call other than `init()` and `initialize()` before it completes the process, the plug-in can generate an `ARException` to notify the caller that it is not ready and to tell the caller its current state. When it is ready, it can process the call.
   - To enable the Java plug-in server to load and instantiate all plug-ins inside the plug-in server, a plug-in should not throw a runtime exception or an `ARException` from the `init()` and `initialize()` methods for nonfatal errors.

7 Add an entry that identifies the plug-in to the plug-in server configuration file. See "Configuring the Java plug-in server."

8 If the Java plug-in uses native libraries, do this:

- Include the native libraries in the `PATH` variable.
- (UNIX only) Include the native libraries in the `LD_LIBRARY_PATH` variable.
- If the libraries need to know the remote host character set, call the `getRemoteHostCharSet()` method of the `ARPluginContext` object. For details, see the Java plug-in API online documentation.

Configuring the Java plug-in server

Configure the Java plug-in server in the `pluginsvr_config.xml` file in the `pluginsvr` subdirectory of your AR System server installation directory. This file should be in the same directory as the plug-in server JAR files and startup script. (It must be in the class path for the plug-in server.) When you add a plug-in to the plug-in server configuration in the `pluginsvr_config.xml` file and save the file, the plug-in server loads the new plug-in after a configured delay. If you remove a plug-in server or make other configuration changes, you must use `armonitor` to stop and restart the plug-in server for it to process the changes.

In AR System 7.5.00, the file encoding of the `pluginsvr_config.xml` file was changed from ISO 8859-1 to UTF-8. This change enables you to use localized plug-in names in the configuration file. If you modify the `pluginsvr_config.xml` file, save it as a Unicode file. Some text editors, such as Windows Notepad, save files as single-byte ANSI (ASCII) by default.

TIP

You should install critical plug-ins, such as an LDAP plug-in that performs LDAP authentication, in separate plug-in servers. If multiple plug-ins are installed on a single server, problems with one plug-in might affect the use of the other plug-ins.

See the sample configuration file, `pluginsvr_config.xml`, for descriptions, valid values, and default values for the Java plug-in server configuration options.

NOTE

See the *Configuration Guide* for configuration of the C-based plug-in server, `arplugin`.

Multithreading in the Java plug-in server

In previous releases, the Java plug-in server created a thread to handle each RPC connection as it was received from the AR System server, often creating many threads. If a connection failed, the plug-in server programmatically shut down the plug-in instances associated with the thread for that connection, often losing data in the process.

To improve performance, the Java plug-in server now uses a configurable pool of worker threads to handle RPC calls. The pool and its associated plug-in instances are created at startup, rather than as RPC calls are received. If a connection fails, the plug-in instances associated with the thread remain instantiated and can still process calls from other connections.

NOTE

Do not send any requests to the Java plug-in server before the plug-ins are loaded and instantiated. (If the plug-in log level is Warn or higher, a message is recorded in the log file when the plug-in server is ready to receive RPC calls.)

When an RPC call is received from the AR Server, a selector thread adds the call to the worker thread task queue. By default, the system uses two selector threads. To prevent bottlenecks, you can increase the number of selector threads by using the `numSelectorThreads` tag in the sample `pluginsvr_config.xml` file.

Whenever a worker thread is free, it starts processing the next request in the task queue. By default, the pool contains five worker threads. To prevent bottlenecks, you can increase the number of worker threads by using the `numCoreThreads` tag in the sample `pluginsvr_config.xml` file.

An unlimited number of tasks can be added to the worker thread task queue. To be notified when too many tasks accumulate in the queue, you can specify a task threshold. When the threshold is exceeded, a message is logged in the `arjavaplugin.log` file. This enables you to avoid potential queue bottlenecks by creating more worker threads in a timely fashion.

To specify the task threshold, use the `workQueueTaskThreshold` tag in the sample `pluginsvr_config.xml` file.

To specify how frequently the system checks to see whether the task threshold has been exceeded, use the `workQueueMonitorLogInterval` tag in the sample `pluginsvr_config.xml` file.

See the plug-in server configuration file (`pluginsvr_config.xml`).

Dynamic plug-in loading

Dynamic plug-in loading is adding or loading a new plug-in definition in the plug-in server without stopping and restarting the BMC Remedy AR System server. In previous releases, most changes made to the `pluginsvr_config.xml` file required a plug-in server restart to take effect.

Java plug-in server

To enable dynamic plug-in loading, you must specify a reload delay before starting the plug-in server (use the `reloadDelay` tag in the sample `pluginsvr_config.xml` file). When a delay is specified, the system automatically loads plug-ins and initiates them for all worker threads *after* the delay period without requiring a plug-in server restart. During the delay period, you can modify the new plug-in configuration if necessary. (If you modify it after the delay, you must restart the plug-in server to make your changes take effect.) For information about restarting the plug-in server, see "Restarting the plug-in server using the Set Server Info command" on page 118.

See the plug-in server configuration file (`pluginsvr_config.xml`).

▶ To configure the Atrium SSO plug-in using only the Java plug-in server

1 Make the following changes in the `ar.cfg` file:

  a Comment out the `Plugin: areaatriumsso.dll` entry (if it exists) and also comment out all the native area plug-in related entries.

  b Modify the `Server-Plugin-Alias` entry for Atrium SSO (if it exists), from `Server-Plugin-Alias: ARSYS.AREA.ATRIUMSSO ARSYS.AREA.ATRIUMSSO iBMC-66r37bs.adprod.bmc.com:9999` to `Server-Plugin-Alias: AREA AREA iBMC-66r37bs.adprod.bmc.com:9999`.

NOTE

Add the entry for `Server-Plugin-Alias` entry if it does not exist for Atrium SSO.

2 Make the following changes in the `pluginsvr_config.xml` file:

  a Change the Atrium SSO plug-in entry from `<name>ARSYS.AREA.ATRIUMSSO</name>` to `<name>AREA</name>`.

3 Restart the AR System server and the Java plug-in server.

NOTE

For more information on Atrium SSO integration, see "Configuring Atrium SSO integration" on page 161.

C plug-in server

Plug-ins are configured in the `ar.cfg` file. A new plug-in is added to the plug-in server through the BMC Remedy AR System server. Previously, adding a new plug-in definition required that you stop and restart the BMC Remedy AR System server. In this release of the BMC Remedy AR System, an API interface in the plug-in server can add the new plug-in definitions dynamically, without stopping and restarting the BMC Remedy AR System server.

See the plug-in server configuration file (`ar.cfg`).

▶ To add the plug-in information to the C plug-in server configuration file

1 Double-click the `driver.exe` file.

The default location of this file is
`C:\Program Files\BMC Software\ARSystem\Arserver\api\driver`.

2 Enter the required login and server information.

3 Enter the `gsi` (Get Server Info) command to get the current plug-in information.

4 Enter the following details:

- Number of server info operations—The number of servers for which you want the current plug-in information.
- Operation—The operation number for getting the current plug-in information that is present in the `ar.cfg` file.

A list of current plug-ins is displayed.

5 Enter the `ssi` (Set Server Info) command to add the new plug-ins.

NOTE

If you enter the `ssi` command without entering the `gsi` command first, the old plug-in entries are overwritten and only the new entries are recorded.

6 Enter the following details:

- Number of server info operations—The number of servers for which the new plug-ins are to be added.
- Operation—The operation number for adding the new plug-ins.
- Datatype—The number for the type of data.

A list of new plug-ins that are added is displayed. The value for the ReturnCode must be OK.

The plug-in information is now updated in the `ar.cfg` file.

Plug-in naming conventions

Plug-in names must be unique. BMC Software, Inc. recommends the following naming format:

`companyName.pluginType.uniquePluginIdentifierName`

For example, if your company name is *ACME*, the type of plug-in is *ARDBC*, and the unique plug-in identifier is `pluginexample1`, your plug-in name could be this:

```
ACME.ARDBC.pluginexample1
```

Plug-in names cannot include spaces or tab characters. In addition, *`uniquePluginIdentifierName`* cannot contain the word *AREA* by itself because that word is reserved for AREA plug-ins. However, you *can* use the word AREA as the *`pluginType`* value.

Restarting the plug-in server using the Set Server Info command

When any changes such as C plug-in or Java plug-in-related changes, binary updates that take place during installation, plug-in related updates to the `ar.conf` file, or changes to the `pluginsvr_config.xml` file are done to the C plug-in or Java plug-in files, you are able to restart only the plug-in server instead of restarting the AR System server.

▶ To restart the plug-in server using the Set Server Info command

1 Double-click the `driver.exe` file.

The default location of this file is `C:\Program Files\BMC Software\ARSystem\Arserver\api\driver`.

2 Enter the required login and server information.

3 (If AR System server is not registered with portmapper) Enter the `ssp` (Set Server Port) command and then enter the server port number.

4 Enter the `ver` command to verify the login information.

5 Enter the `ssi` (Set Server Info) command to then enter the number of server info operations that you want to perform.

6 Enter the operation number (value is 348) to forcefully shut down the plug-in server.

7 Enter the data type as Integer (value is 2) and the integer value.

When the AR monitor detects that the plug-in server is down, it checks if any changes are made to the `ar.cfg` file. If the changes are detected, the recent `ar.cfg` is loaded before the stopped plug-in server is automatically restarted.

Configuring the AR System server

To access a plug-in server, the AR System server needs its host name and port number. The server searches for the host name and port number in this order:

1 The plug-in alias, if any.

2 The plug-in server entry on the Connection Setting tab of the AR System Administration: Server Information form. (This entry is also required to set a password for a plug-in server.)

3 The local host and the port number specified by the `Plugin-Port` setting in the `ar.cfg` or `ar.conf` file. (See the *Configuration Guide* for information about the AR System Administration: Server Information form and the server configuration file.)

4 The port number that the plug-in server registered with the portmapper.

To access one C or Java plug-in server with no password running on the same computer as the AR System server, only the `Plugin-Port` option in the `ar.cfg` or `ar.conf` file is required. To specify a password for a plug-in server, an entry on the Connection Setting tab of the AR System Administration: Server Information form is required.

To access two or more plug-in servers, for example, to access both the C and Java plug-in servers or to access plug-in servers on two or more computers, define aliases for all plug-ins other than those loaded by the primary plug-in server that is set up as described in the previous paragraph.

Defining plug-in aliases

You define plug-in aliases in the `ar.cfg` or `ar.conf` configuration file. Use this format:

```
Server-Plugin-Alias: aliasName realName hostName[:portNumber]
```

Parameter	Description
aliasName	Name referenced in AR System applications. AR Filter API calls and vendor forms reference this alias name. This is an arbitrary string, but it cannot include semicolons or blank-space characters, such as spaces, tabs, or new lines.
realName	Actual name that the plug-in exposes to the plug-in server.
hostName	Name of the host the AR System server accesses to find the associated plug-in server.
portNumber	Port number the AR System server connects with when accessing the associated plug-in server. This is optional. If you do not specify a port number, the `Plugin-Port` is used.

Examples of plug-in aliases

Following are examples of how aliases affect the behavior of the AR System server when it accesses plug-ins.

Example 1

```
Server-Plugin-Alias: RMDY.ARDBC.XML RMDY.ARDBC.XML myhost
```

A vendor form that accesses the ARDBC plug-in named `RMDY.ARDBC.XML` is redirected to the plug-in by the same name on the plug-in server running on myhost.

Example 2

```
Server-Plugin-Alias: RMDY.ARF.PERL.myhost RMDY.ARF.PERL myhost
```

Workflow that accesses the ARF plug-in named `RMDY.ARF.PERL.myhost` is redirected to the `RMDY.ARF.PERL` plug-in on the plug-in server running on myhost.

Example 3

```
Server-Plugin-Alias: RMDY.ARDBC.LDAP.fred RMDY.ARDBC.LDAP
myhost:11001
```

A vendor form that accesses the ARDBC plug-in named `RMDY.ARDBC.LDAP.myhost.1` is redirected to the `RMDY.ARDBC.LDAP` plug-in on the plug-in server running on myhost and listening at port number 11001.

Example 4

```
Server-Plugin-Alias: AREA AREA myhost
```

When the AR System server accesses the AREA plug-in, it connects to the plug-in on the plug-in server that is running on `myhost`. Only one AREA plug-in can exist, so use the reserved name `AREA` for the alias.

Running the plug-in server

The plug-in server is set up in the `armonitor` configuration to start automatically at system startup. It stops and restarts with the rest of the AR System services controlled by `armonitor`, the AR System UNIX daemon, or the Windows service.

To start the plug-in server manually, run the `pluginsvrstartup` command in the `pluginsvr` directory. This command file (`pluginsvrstartup.sh` for UNIX or `pluginsvrstartup.bat` for Windows) is customized for your installation. To change command-line options, see the description of the Java plug-in server in the list of AR System components in the *Configuration Guide*.

Logging plug-in information

Plug-ins can write information to the plug-in server log file. C plug-ins can use the ARPluginSetProperties function to call the plug-in log function:

```
typedef int (*AR_PLUGIN_LOG_FUNCTION)(
  ARPluginIdentification      *id,
  int                         logLevel,
  char                        *text);
```

Argument	Description
id	The plug-in type, name, and version.
logLevel	The log level to which the information applies: ■ AR_PLUGIN_LOG_OFF (10000)—No plug-in messages are logged. (Some plug-ins may ignore this setting.) ■ AR_PLUGIN_LOG_SEVERE (1000)—Messages that report fundamental problems that prevent a plug-in from working (for example, the inability to open a required resource during plug-in initialization). ■ AR_PLUGIN_LOG_WARNING (900)—Messages that might be precursors to severe problems or identify incorrect configuration settings (for example, for a bad configuration setting, a plug-in might log a warning and reverts to a default value). ■ AR_PLUGIN_LOG_INFO (800)—Messages that identify intermittent milestones or events that do not have negative repercussions. This should not be used for information that is likely to occur frequently. ■ AR_PLUGIN_LOG_CONFIG (700)—Messages that describe the current configuration settings of the plug-in. ■ AR_PLUGIN_LOG_FINE (600)—Messages that identify the result of every decision made throughout processing. This level, along with the FINER and FINEST log levels, is primarily used while resolving problems. ■ AR_PLUGIN_LOG_FINER (500)—Supporting data that supplements FINE messages. ■ AR_PLUGIN_LOG_FINEST (400)—Messages that contain information to help users who have plug-in source code in front of them. At this level, messages can reference internal function names or structures. (All messages logged at higher levels should have meaning for users who might not have access to the source code.) ■ AR_PLUGIN_LOG_ALL (100)—All plug-in messages are logged. You can specify a log level for each situation. This enables the plug-in to write different information to the log file depending on the log level configured for the plug-in server The information is not written to the log file unless the plug-in server log level is equal to or lower than the value of logLevel.
text	The message that is written to the plug-in server log file.

The C plug-in log function has no return value.

Set the log level for the C-based plug-in server using the Plugin_Log_Level option in the ar.cfg or ar.conf file or on the Log Files tab of the AR System Administration: Server Information form. For more information, see the *Configuration Guide*, "Server Information—Log Files tab," page 150.

Java plug-ins can log messages using the logMessage method of their ARPluginContext object. See the Java plug-in API online documentation for details.

The Java plug-in server uses the log4j utility. Set the log level and other logging configuration in the log4j_pluginsvr.xml file. Comments in the sample file describe the log configuration options.

Logging exceptions for calls to Java plug-ins

When a run-time exception or an `ARException` class error occurs during a Java plug-in server call to a Java plug-in, the following information is now recorded in the `*ARServerInstallDir*\Arserver\Db\arjavaplugin.log` file:

- The name of the plug-in

  This name matches the name of the corresponding plug-in library registered in the Java plug-in configuration file, `pluginsvr_config.xml`. For example, if the library is registered as `<name>ARSYS.ARF.WEBSERVICE</name>`, the plug-in name in the log file is `ARSYS.ARF.WEBSERVICE`.

- The method that the server tried to call in the plug-in
- (Run-time exceptions only) The exception stack trace

In addition, when a run-time exception occurs, users receive `Error 8753: Error in plugin:` *`pluginName`*.

When an `ARException` occurs, users receive the usual message associated with the exception.

NOTE

When you restart the AR System server, the arjavaplugin.log might log warnings that look similar to this:

```
2009-10-14 11:07:12,573 WARN  [pool-2-thread-1]
com.bmc.arsys.pluginsvr.plugins.ARPluginContext (?:?) -
<ARSYS.ARF.REGISTRY>Null registry location
```

You can safely ignore these messages. If you want to use the Registry, then enter the Registry location in the AR System Administration Console. For more information, see “Registering a web service” on page 80.

C plug-in exception handling

Exception handling in the C plug-in server now produces a stack trace. The stack trace includes the names of the operation, vendor, and plug-in library. It is written to the `arerror.log` file.

Common plug-in C functions and Java methods

This section describes the calls or methods common to all plug-in types.

WARNING

Plug-in operations run synchronously; that is, AR System waits for a plug-in to complete its processing before the server continues its processing. Thus, a badly written plug-in can adversely impact AR System performance and stability. A plug-in developer must (1) maintain thread safety, (2) use minimal processing logic for optimal code execution, (3) implement only callback methods that are required or that have custom logic for a given plug-in, (4) allocate or free resources appropriately to prevent resource leaks, and (5) optimize any costly operation that must be performed (or data structures that must be built) by employing a meaningful caching strategy.

Common C plug-in functions

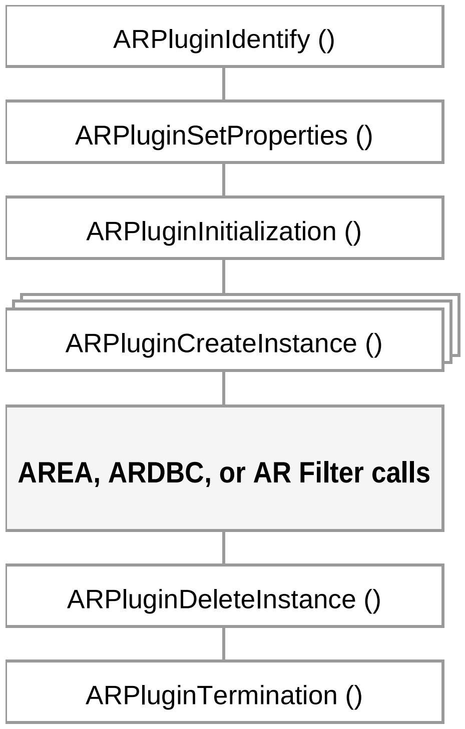
The following C API functions are common to every type of plug-in:

- `ARPluginCreateInstance`
- `ARPluginDeleteInstance`
- `ARPluginEvent`
- `ARPluginIdentify`
- `ARPluginInitialization`
- `ARPluginSetProperties`
- `ARPluginTermination`

In general, these functions use the same structure as other AR System C API functions. The plug-in server calls the functions and provides the necessary input data. The plug-in accepts the data, processes it, and returns the appropriate response data to the plug-in server.

Figure 7-2 shows a typical sequence of calls that the plug-in server makes on a C plug-in.

Figure 7-2: C plug-in call sequence


ARPluginIdentify ()
ARPluginSetProperties ()
ARPluginInitialization ()
ARPluginCreateInstance ()
AREA, ARDBC, or AR Filter calls
ARPluginDeleteInstance ()
ARPluginTermination ()

For more information, see the *C API Reference*.

Common Java plug-in methods

The methods defined in the `ARPluggable` interface and the `ARPlugin` abstract class are common to all plug-in types. For more information, see the Java plug-in API online documentation.

AREA plug-ins

AR System External Authentication (AREA) provides a way to validate users by connecting AR System to a data source outside the AR System database. This can be done using the AREA LDAP plug-in or by creating your own custom plug-in for authentication services such as Kerberos. See "Creating C plug-ins" on page 110 and "AREA plug-in C API functions" on page 126 for details.

When users first log in to AR System through a client or when a client issues an API call to AR System, the AR System server verifies the user name and password.

If the server verifies that the user name and password are in the User form, it authenticates the information and processes the login or API call.

If the user information is not in the User form or if the user password is blank in the User form, the AR System server sends an authentication request to the plug-in server. The request passes from the plug-in server through the AREA plug-in instance to the external authentication source. The external authentication source sends authentication information back through the same path to the AR System server.

NOTE

For the AR System server to use an AREA plug-in to authorize logins, the corresponding entries in the User form must have blank passwords.

If the authentication source verifies that the user information is valid, the AR System server processes the API call or allows the user to log in.

When the authentication information is not verified (that is, the information is incorrect, incomplete, or cannot be found in the external data source), the AR System server returns an error message to the client.

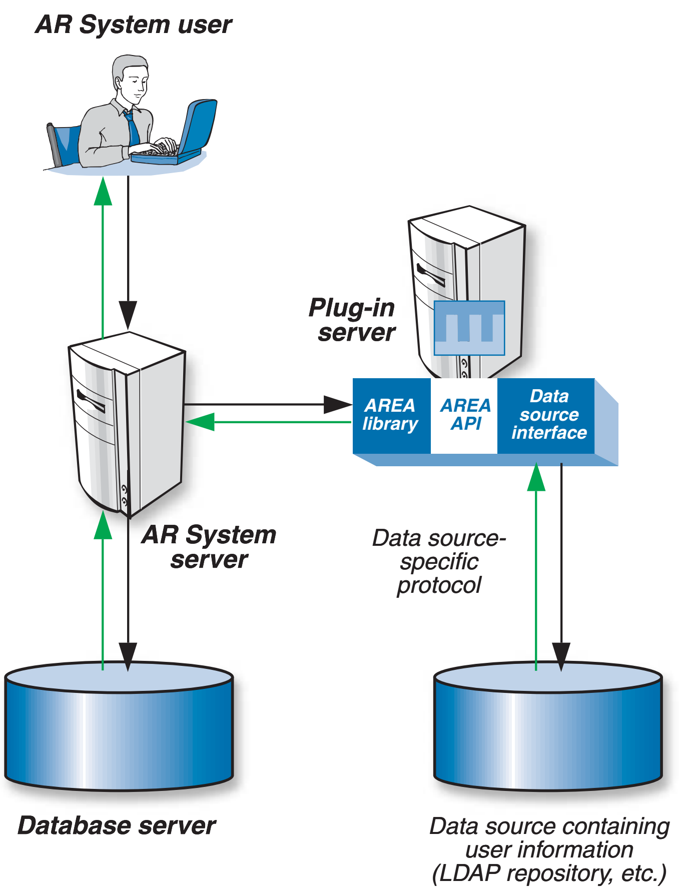
The plug-in can load only one AREA plug-in instance at a time. An AREA plug-in can be configured to access one or more data sources.

AREA plug-ins can selectively override field values entered in the User form.

NOTE

The plug-in behavior depends on how you configure the plug-in, such as whether you enable the Cross Reference Blank Password and the Authenticate Unregistered users options.

Figure 7-3: External authentication architecture



AREA plug-in C API functions

These are the AREA plug-in API functions:

- `AREAFreeCallback`
- `AREANeedToSyncCallback`
- `AREAVerifyLoginCallback`

For more information, see the *C API Reference*.

AREA plug-in Java API methods

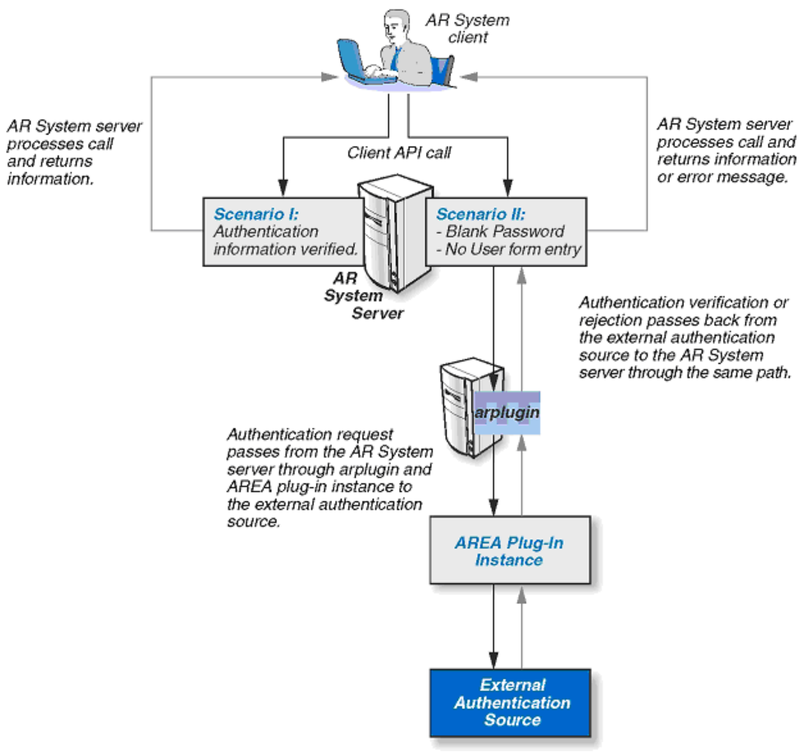
The methods defined in the `AREAPluggable` interface and the `AREAPlugin` abstract class are common to all plug-in types. For more information, see the Java plug-in API online documentation.

Sample AREA implementations

When you install AR System, you can install a sample C AREA LDAP implementation, including an AREA LDAP plug-in. That plug-in provides you with an integration point between AR System and LDAP directory services.

You must create a custom plug-in to integrate AR System with external authentication services such as Kerberos. See "Creating C plug-ins" on page 110 and "AREA plug-in C API functions" on page 126 for details.

Figure 7-4: Example flow of requests and data for an AREA plug-in


AR System client
AR System server processes call and returns information.
Client API call
AR System server processes call and returns information or error message.
Scenario I: Authentication information verified.
Scenario II: - Blank Password - No User form entry
AR System Server
Authentication verification or rejection passes back from the external authentication source to the AR System server through the same path.
arplugin
Authentication request passes from the AR System server through arplugin and AREA plug-in instance to the external authentication source.
AREA Plug-In Instance
External Authentication Source

ARDBC plug-ins

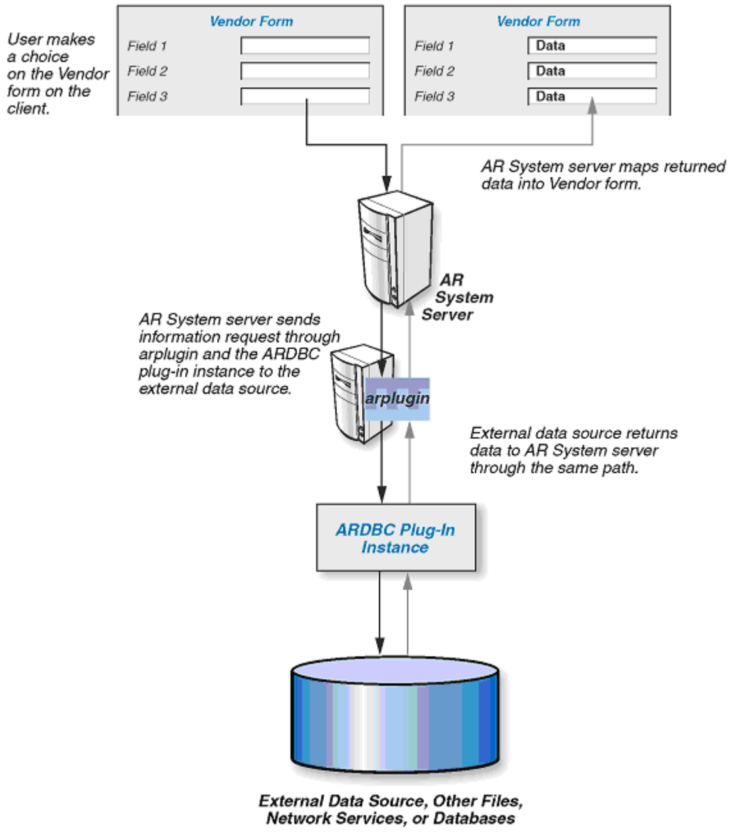
AR System Database Connectivity (ARDBC) plug-ins provide access to data stored outside the AR System database as if it were located in tables owned by AR System. After an ARDBC plug-in is installed, the AR System administrator can create a vendor form that references the table and columns of the external data source. Views and workflow can then be built for vendor forms just as if they were regular AR System forms. The source of data manipulated by the AR System API client, such as BMC Remedy User or BMC Remedy Mid Tier, is transparent to the user.

When users enter requests in the vendor form, the AR System server sends the requests to the plug-in server, which sends the requests to the ARDBC plug-in instance. The plug-in retrieves the data (if any) from the external data source and returns it in the opposite direction. The AR System server maps the external data to fields in the vendor form, and the form displays the data. See Chapter 11, "Vendor forms."

Unlike AREA plug-ins, multiple ARDBC plug-ins can be loaded simultaneously by the plug-in server.

Figure 7-5 shows the flow of requests and information for an ARDBC plug-in.

Figure 7-5: Flow of requests and information for the ARDBC plug-in


User makes a choice on the Vendor form on the client.
Vendor Form
Field 1
Field 2
Field 3
Vendor Form
Field 1
Data
Field 2
Data
Field 3
Data
AR System server maps returned data into Vendor form.
AR System Server
AR System server sends information request through arplugin and the ARDBC plug-in instance to the external data source.
arplugin
External data source returns data to AR System server through the same path.
ARDBC Plug-In Instance
External Data Source, Other Files, Network Services, or Databases

ARDBC plug-in C API functions

An ARDBC plug-in API enables AR System to:

- Implement calls on an external data source that are analogous to set entry, get entry, create entry, delete entry, and get list C API calls.
- Use external data to implement Push Field and Set Field filter, escalation, and active link actions.
- Create, modify, and search for external data through API clients, such as BMC Remedy User.

- Construct query-style character menus.
- Present forms, views, and active links on external data in the same manner as internal data. The data source is transparent to the user.

When you create an ARDBC plug-in, be sure to completely document the capabilities of your plug-in. For example, you might create a read-only plug-in, which does not allow a user to create, set, or delete entries.

If the data source with which you integrate does not support a particular functionality, do not implement that function. Instead, let the default behavior occur. For example, if your data source does not support transactions, do not define `ARDBCCommitTransaction` and `ARDBCRollbackTransaction` functions.

These are the ARDBC plug-in API functions:

- `ARDBCCommitTransaction`
- `ARDBCCreateEntry`
- `ARDBCDeleteEntry`
- `ARDBCGetEntry`
- `ARDBCGetEntryBLOB`
- `ARDBCGetEntryStatistics`
- `ARDBCGetListEntryWithFields`
- `ARDBCGetListSchemas`
- `ARDBCGetMultipleFields`
- `ARDBCRollbackTransaction`
- `ARDBCSetEntry`

For more information, see the *C API Reference*.

Calling AR System API from an ARDBC plug-in

A C ARDBC plug-in can make AR System C API calls to the AR System server. In pre-7.0 versions of AR System, such calls had to be made with a known user account. Now, you can make the calls as the same user whose operation led to the ARDBC plug-in call. This ensures that any call from the ARDBC plug-in has the same permissions as the user who called the ARDBC plug-in.

When a plug-in call is made, AR System server creates a globally unique ID (GUID) to identify the user instance calling the plug-in. The plug-in provides callback routines to fetch the user name, authentication string, and GUID. Subsequently, when a plug-in makes an API call, it uses those callback routines to fetch the information it needs to authenticate itself as the user that made the original call to the plug-in.

The calling plug-in uses the following API calls to set the callback routines for the API to fetch the user name, authentication string, and authenticating GUID:

- `ARSetUserNameSource`
- `ARSetAuthStringSource`
- `ARSetNativeAuthenticationSource`

Pointers to the callback routines are made available to the plug-ins as members of a properties list (`ARPropList`) passed as an argument to `ARDBCPluginSetProperties` (if implemented by the plug-in) when the plug-in is loaded. The plug-in must save these pointers and use them later as arguments to API calls. These API calls must be made immediately after the `ARIntitialization` call before any other API calls.

NOTE

When using the GUID authentication feature from a plug-in, internal users (such as ESCALATOR and ARCHIVE) encounter errors. The errors occur because these users are not valid users for making API calls.

For more information, see the *C API Reference*.

A Java ARDBC plug-in can make AR System Java API calls to the AR System server. Use the `ARPluginContext` object to create a `ARServerUser` object. See the Java plug-in server API online documentation.

ARDBC plug-in Java API methods

The methods defined in the `ARDBCPluggable` interface and the `ARDBCPlugin` abstract class are common to all plug-in types. For more information, see the Java plug-in API online documentation.

Creating a vendor form for an ARDBC plug-in

After you build and install an ARDBC plug-in and configure your server to recognize it, you can create a vendor form. For information about configuring your server to recognize a plug-in, see the *Configuration Guide*.

NOTE

Creating a vendor form for an ARDBC LDAP plug-in is a special case. See "Creating a vendor form to represent a collection of LDAP objects" on page 140.

▶ To create a vendor form for an ARDBC plug-in

1. In BMC Remedy Developer Studio, choose File > New > Vendor Form.
2. In the New Vendor Form Wizard, select the server on which you want to create the vendor form and click Next.
3. Select the ARDBC plug-in to use in the list of Available Vendor Names, and click Next.

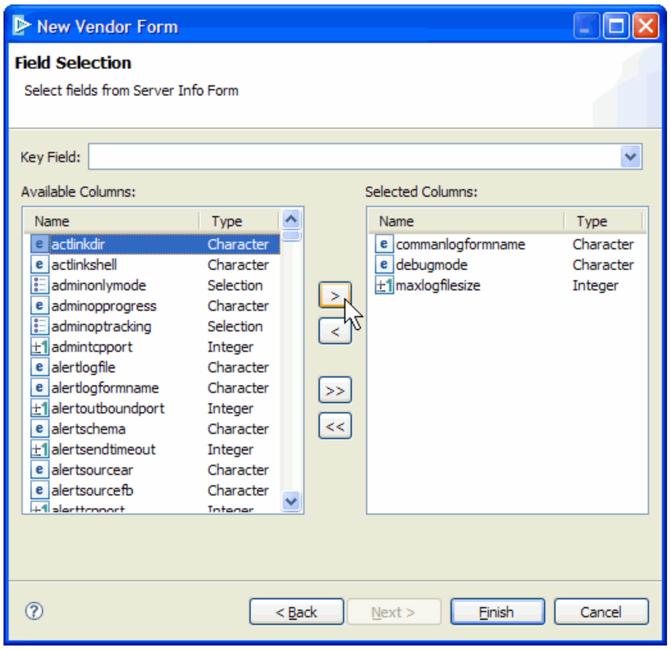
4 Choose a table from the list of Available Vendor Tables, and click Next.

Alternatively, type a table name in the Table field, click Validate, then click Next.

5 (optional) On the Field Selection page, choose a key column in the Key Field list box.

6 In the Available Columns list on the Field Selection page, select columns to access in AR System. Use the arrow buttons to move them to the Selected Columns list.

Figure 7-6: New Vendor Form Wizard, Selected Columns



Available Columns:

Name	Type
actlinkdir	Character
actlinkshell	Character
adminonlymode	Selection
adminopprogress	Character
adminoptracking	Selection
admintcpport	Integer
alertlogfile	Character
alertlogformname	Character
alertoutboundport	Integer
alertschema	Character
alertsendtimeout	Integer
alertsourcear	Character
alertsourcefb	Character
alerttcpport	Integer

Selected Columns:

Name	Type
commanlogformname	Character
debugmode	Character
maxlogfilesize	Integer

7 Click Finish to create the vendor form.

8 Use Developer Studio to edit the new form, then click File > Save.

Issues and considerations

Keep these issues in mind when creating a vendor form:

- The plug-in can load more than one ARDBC plug-in at a time.
- Full Text Search (FTS) operations are not available on vendor form fields.
- You can add only those Required and Optional fields that correspond to actual columns in the data source. In addition, you can add a Display Only field only when the column name does not correspond to a column in the data source.

For more information about vendor forms, see the *Form and Application Objects Guide*.

AR filter API plug-ins

AR System filter (AR filter) API plug-ins enable you to create tight, efficient integrations between your systems and the AR System server. They are triggered with Set Field actions in filters and escalations. Because this middleware is loaded as a plug-in when AR System is started instead of as a standalone executable at each event, it consumes fewer resources and less processor time than a Set Fields `$PROCESS$` action.

You use BMC Remedy Developer Studio to create AR Filter Set Field actions in filters and escalations.

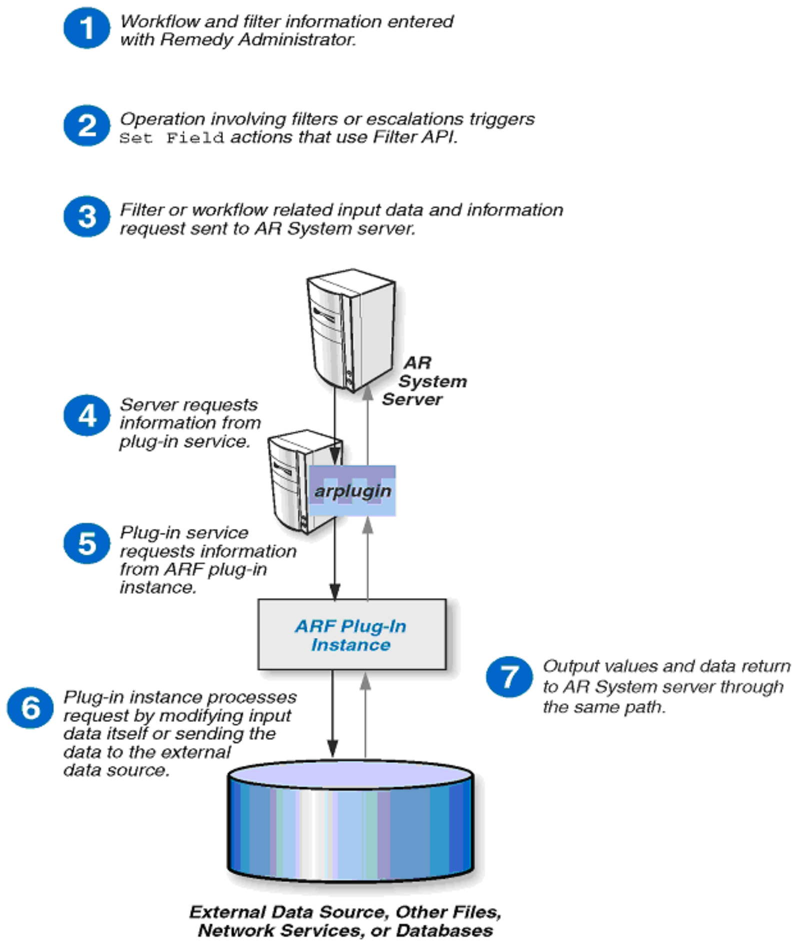
At run time, the AR System server sends AR filter API requests to the plug-in, which directs the requests to the appropriate plug-in. The plug-in processes the input arguments and can return values that can be used in the Set Fields action.

When you enter AR Filter API requests in the Create Filter form, the AR System server sends the requests to the plug-in, which sends them to AR Filter. AR Filter either processes the data or request itself or retrieves output data from the external data source and returns it in the opposite direction.

Unlike AREA plug-ins, multiple AR filter API plug-ins can be loaded simultaneously by the plug-in.

Figure 7-7 shows the flow of requests and information for AR filter API plug-ins.

Figure 7-7: Flow of requests and information for the AR filter API plug-in


1 Workflow and filter information entered with Remedy Administrator.
2 Operation involving filters or escalations triggers Set Field actions that use Filter API.
3 Filter or workflow related input data and information request sent to AR System server.
AR System Server
4 Server requests information from plug-in service.
arplugin
5 Plug-in service requests information from ARF plug-in instance.
ARF Plug-In Instance
7 Output values and data return to AR System server through the same path.
6 Plug-in instance processes request by modifying input data itself or sending the data to the external data source.
External Data Source, Other Files, Network Services, or Databases

AR filter API plug-in C function

The AR filter API plug-in API has one function, `ARFilterApiCall`. For more information, see the *C API Reference*.

The AR filter API plug-in function is a blocking call, so the AR System server thread that makes the call waits for the plug-in to respond. For best performance, the plug-in should return quickly. Tell your plug-in installers about the expected latency, and have them set their `AR_SERVER_INFO_FILTER_TIMEOUT` value accordingly.

The following example files, which can be used to create an AR Filter API DLL or shared library, are located in the `*ARSystemServerInstallDir*/Arserver/Api/arfilterapi/example` directory:

- `arfilterapisamp.c`
- `arfilterapisamp.dep`
- `arfilterapisamp.vcproj`
- `arfilterapisamp.mak`

AR filter API plug-in Java methods

The methods defined in the `ARFilterPluggable` interface and the `ARFilterPlugin` abstract class are common to all plug-in types. For more information, see the Java plug-in API online documentation.

Chapter 8 LDAP plug-ins

This section describes how to configure and use the ARDBC and AREA LDAP plug-ins to integrate AR System with a directory service.

The following topics are provided:

- Overview of LDAP and AR System (page 136)
- ARDBC LDAP plug-in (page 136)
- AREA LDAP plug-in (page 145)

Overview of LDAP and AR System

Lightweight Directory Access Protocol (LDAP) provides a standard method for accessing information from a central directory. A common use for LDAP is user authentication. After a user is set up in the LDAP directory, he or she can use the same user name and password to log in to any application that supports the LDAP protocol.

AR System provides these LDAP plug-ins:

- **AR System Database Connectivity (ARDBC) LDAP**—Accesses data objects stored in a directory service as if they were entries stored in a typical AR System form. You can search for, modify, and create data in a directory service using this plug-in. You can also use the data in workflow and to populate character menus and table fields. See "ARDBC LDAP plug-in" on page 136.
- **AR External Authentication (AREA) LDAP**—Authenticates AR System users against external LDAP directory services. See "AREA LDAP plug-in" on page 145.

ARDBC LDAP plug-in

The AR System Database Connectivity (ARDBC) LDAP plug-in enables you to access data from an external LDAP system through vendor forms.

Vendor forms are AR System objects that present external data as entries in an AR System form. Using vendor forms and the ARDBC LDAP plug-in, you can view and manipulate external LDAP data as if it were stored in the AR System database. See Chapter 11, "Vendor forms."

Requirements

When using the ARDBC LDAP plug-in, remember these facts:

- The ARDBC LDAP plug-in uses the LDAP v3 protocol to issue requests to directory services.
- Attributes of directory service objects consist of either character data, integer data, or time stamps. Attachments are not supported.
- By default, all attributes have one value. Multivalued attributes are supported by a special notation. See "Multivalued attributes" on page 143.
- LDAP does not support transactions. Consequently, when an object is created, modified, or deleted, the change does not roll back if subsequent workflow in the AR System server detects an error condition.
- The distinguished name and password specified in the ARDBC LDAP configuration are used to connect to any directory service referenced in LDAP search URLs.
- Only server-based certificates are supported.

Configuring the ARDBC LDAP plug-in

You must configure the ARDBC LDAP plug-in *before* you create the vendor form used to access user information in your particular LDAP server.

You configure ARDBC through parameters in the `ar.cfg` file and the properties of the vendor form. To make configuration easier, the installation of the ARDBC LDAP plug-in adds these forms:

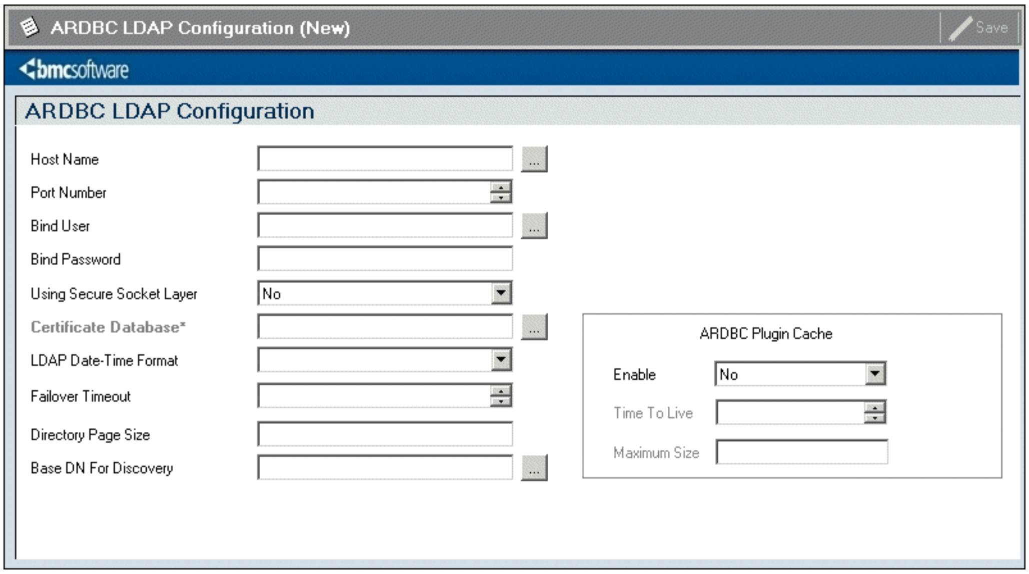
- **ARDBC LDAP Configuration**—A vendor form, using a separate plug-in, that reads and writes to the `ar.cfg` file.
- **Configuration ARDBC**—A display-only form that uses filters to push values to the Configuration ARDBC Form and, as a result, to the `ar.cfg` file.

▶ To configure the ARDBC LDAP plug-in

1 In the AR System Administration Console, click System > LDAP > ARDBC Configuration.

The ARDBC LDAP Configuration form opens in New mode.

Figure 8-1: ARDBC LDAP configuration form



2 In the Host Name field, enter one or more host names of the directory service from which you want information for the vendor form. You can specify a space-separated list of host names up to 255 characters long. Starting with the first host name in the list, AR System tries to connect to each server until it is successful.

If you use Secure Socket Layer (SSL), this host name should match the name for which the server's certificate was issued.

3 In the Port Number field, enter a port number for this directory service. The default port number is 389. (For an SSL connection, the default is 636.)

4 In the Bind User field, enter the distinguished name of the user account that the ARDBC LDAP plug-in uses to log in to the directory service. The administrator who set up the LDAP service designated this name. With the vendor form, some LDAP servers allow you to make an anonymous connection. If you plan to use an anonymous connection, leave the Bind User and Bind Password fields blank.

   Otherwise, use a standard distinguished name such as `cn=manager, dc=remedy, dc=com`.

5 In the Bind Password field, enter the password for the user account. (For security, asterisks replace the characters you enter for the password.)

   If you leave the Bind Name and Password fields blank, you are connected anonymously.

6 To use a Secure Socket Layer (SSL) connection, select Yes in the Using Secure Socket Layer field; otherwise, accept the default value No. If you select Yes, the Certificate Database field becomes active, and you can enter a certificate database as described in step 7. Because SSL requires additional setup in this form and outside AR System, you might first want to experiment without SSL and then add this option later.

7 In the Certificate Database field, enter the path to the directory containing the certificate database file. Do not include the file name in the path.

   To create a certificate database, see "Adding a certificate to a certificate database" on page 401.

8 In the LDAP Date-Time Format field, select the format to use to represent date and time to LDAP servers.

Format	**Value**	**Description**	**Example: 6 a.m. Sept. 28, 2001**
Generalized Time	0	`YYYYMMDDHHMMSSZ` This format is recognized by all LDAP servers, and it is recommended.	`20010928060000Z`
AD Generalized Time	1	`YYYYMMDDHHMMSS.0Z` This format is recognized only by Microsoft Active Directory servers.	`20010928060000.0Z`
UTC Time	2	`YYMMDDHHMMSSZ` This is a historical format and does not indicate the century. It is not recommended.	`010928060000Z`

   For more information, see the *Configuration Guide*.

9 In the Failover Timeout field, specify the number of seconds in which the directory service must respond to the plug-in server before an error is returned. The minimum value is `0` (which means the connection must be made immediately). The failover time-out cannot be set higher than the value of the `Server-Plugin-Default-Timeout` parameter.

10 In the Directory Page Size field, enter the number of entries to return in a single page to the client from the external directory server when a search request is processed.

TIP

Setting the Directory Page Size to 1000 can help improve your system's performance while you design and create vendor forms.

11 In the Base DN For Discovery field, enter a base distinguished name to use instead of the root distinguished name as the basis for obtaining the list of vendor tables.

TIP

Specifying a value in the Base DN For Discovery field can help improve your system's performance while you design and create vendor forms.

12 In the ARDBC Plugin Cache box, specify this ARDBC plug-in caching information:

a In the Enable field, select Yes to enable ARDBC plug-in caching.

b In the Time To Live field, specify how long data should be kept in the ARDBC plug-in cache.

c In the Maximum Size field, specify the maximum size of the cache.

TIP

Enabling the ARDBC plug-in cache can help improve your system's performance at run time.

13 Click Save.

The system updates the `ar.cfg` (`ar.conf`) file with the parameters you specified in this form.

For more information, see the *Configuration Guide*.

Building AR System forms for directory services

This section describes the concepts and generic procedures involved in building vendor forms and workflow in AR System to access data stored in a directory service. The following topics are covered:

- "Organizing data"
- "Creating a vendor form to represent a collection of LDAP objects" on page 140
- "Identifying objects uniquely" on page 141
- "Supporting object creation" on page 142

Organizing data

Data in a directory service is organized differently from traditional database applications. Traditional database applications organize data in tables that have a fixed number of columns. Each row in a table represents a single entity and contains a value for each column in the table.

A directory service organizes data as a collection of objects. Each object is characterized by one or more object classes that define the values, or attributes, that the object defines. In addition, objects might be grouped into organizational units.

Because of these differences, there is no one-to-one mapping between rows / columns / tables and objects / attributes / object classes. The following table shows relationships among directory service, AR System, and typical database concepts.

Directory service	AR System	Database
Object class	Form	Table
Attributes	Field	Column
Object	Entry	Row

The following sections describe how to map the directory service and AR System data sources.

Creating a vendor form to represent a collection of LDAP objects

A table in a database can be described as a collection of rows. The data associated with an AR System form is described as a collection of entries.

A collection of objects in a directory service is similar to entries in a form or a collection of rows in a table. When working with a directory service, you can use a standard LDAP search URL to describe a collection of objects. An LDAP search URL looks something like this:

```
ldap://orangina/o=remedy.com??sub?(objectclass=inetorgperson)
```

This URL contains the components described in the following table.

URL component	Description
`ldap://`	LDAP protocol.
`orangina`	Directory service host name.
`o=remedy.com`	Search base name.
`sub`	Search scope. In this case, `sub` indicates that the search applies to the entire subtree under the base name.
`(objectclass=inetorgperson)`	Filter that selects the objects. Note: You should define the search URL to retrieve objects that inherit from a particular object class. You should not mix unrelated objects (for example, people and computers). They might have different sets of attributes, making the search difficult to manage and administer.

NOTE

The LDAP URL standard enables you to specify a list of attributes to be returned by the search. This attribute list would ordinarily fall between the base name and search scope in the URL. In the previous example, no attributes are listed because the LDAP plug-in ignores the attribute list. Instead, you identify attributes in the field properties. See "Alternative method of adding a field to represent the uid (User ID) attribute" on page 366.

Each object selected by the LDAP search can be represented as an entry in a vendor form. You use BMC Remedy Developer Studio to create a vendor form and add fields to which you attach LDAP data.

By default, the AR System server returns the Short Description field (field ID 8) in a results list. Because vendor forms do not have a Short Description field, so you must define the fields that will appear in the results list. Include the vendor form's key field in the results list so that each record is uniquely identified. For more information, see *Form and Application Objects Guide*, "Defining search results," page 178.

For general information about creating vendor forms, see Chapter 11, "Vendor forms."

For an example of how to create a vendor form for LDAP data, see Appendix C, "ARDBC LDAP example: Accessing inetorgperson data."

Identifying objects uniquely

AR System uniquely identifies entries in a form through the Request ID field.

Objects in a directory service have an attribute called the distinguished name (`dn`), which uniquely identifies each object. An object's distinguished name often consists of one attribute or multiple concatenated attributes. For example:

```
uid=abarnes,ou=People,o=remedy.com
```

AR System Request IDs are 15 bytes maximum in length and are assigned by AR System when the entry is created. Distinguished names, on the other hand, are often longer than 15 bytes. However, you can map distinguished names longer than 15 bytes to AR System Request IDs.

When designing an AR System form to access data stored in a directory service, you must determine what attribute to use to distinguish one object from another.

NOTE

If you specify an attribute for the Request ID that resolves to an empty value for an object in the directory service, you receive an `ARERR (100) Entry ID list is empty` message, and no records are displayed in the client. If more than one record has the same value, you retrieve data only for the first matching entry.

For example, in a typical system the `dn` attribute uniquely identifies objects defined by the `inetorgperson` object class. You would create a field for User ID and associate *both* the Request ID field and the User ID field with the `dn` attribute.

NOTE

This is the only case where you should associate one attribute with more than one AR System field. Associating an attribute with more than one field might lead to run-time errors or incorrect behavior.

Supporting object creation

This section describes how to use the ARDBC LDAP plug-in to create objects in the directory service. To support the creation of objects by using the ARDBC LDAP plug-in, you must perform the following tasks:

- Create an AR System field that is associated with the `objectclass` attribute. The `objectclass` attribute is a multivalue attribute.
- Create a field (other than Request ID) associated with `dn`, and define workflow that assigns a value to it. Although entries in AR System are uniquely identified by the Request ID, objects in a directory service are uniquely identified by the `dn` (distinguished name) attribute.
- Add any attributes that are required to your AR System form. Many object classes require that you specify values for certain attributes. These are similar to required fields in AR System.

Creating an objectclass field

`objectclass` is a multivalued attribute that describes all object classes from which an object inherits. Each object class defines a set of attributes. If an object inherits from an object class, it can have values for those attributes. An object can inherit from more than one object class; therefore, an object can have values for all combined attributes.

When you create an object, you must specify all the object classes from which the object inherits. You must add a character field to your form, attach this field to the `objectclass` attribute, and use the multivalue attribute notation (see "Multivalued attributes").

Because all objects associated with an AR System form belong to the same object classes, you can easily set the default value of the field to the object class list. For example, the default value for the object class field associated with an `inetorgperson` object class is this:

```
top,person,organizationalperson,inetorgperson
```

The `inetorgperson` class inherits from the `top`, `person`, and `organizationalperson` classes.

Because the value does not change for this field, you should make this field Read Only. You might also want to make the field Hidden.

Multivalued attributes

Most attributes in an object class are defined to support one value. Some attributes, however, can have many values. For example, a "person" object includes a "telephone number" attribute that allows you to specify many phone numbers. When this attribute is retrieved, the directory service can return any number of telephone numbers as atomic values.

This differs from typical database applications and AR System, in which a column or field stores only one value. To store two phone numbers in such applications, you must add a new column or field to accommodate the additional data.

To resolve this difference between the two data models, use a *special notation* when specifying the attribute name in the Field Properties window:

```
attributeName[*separatorString]
```

Values associated with *`attributeName`* are concatenated into a single value in AR System but separated with *`separatorString`*. For example, to concatenate all values associated with the `telephoneNumber` attribute and separate each value with a comma you would enter the following as the attribute name in the Form Properties window:

```
telephoneNumber[*,]
```

You could then define workflow to extract, add, or modify values in the comma-separated list of telephone numbers.

Generating and assigning a distinguished name

The distinguished name (`dn`) attribute is generally assigned a value through workflow. The workflow takes one or more values and assembles the values for the `dn` attribute. After the `dn` is assigned at creation, it typically does not change just as the Request ID does not change in an entry under an AR System form.

This is done using a filter that executes on a submit operation. You define the filter to perform a Set Fields operation. For more information about creating filters, see the *Workflow Objects Guide*.

ARDBC LDAP run-time performance tips

You can improve your ARDBC LDAP run-time performance by using time-based queries and caching.

Time-based queries

Time-based queries reduce the time it takes to search your directory service.

AR System retrieves `modifyTimestamp` and `whenChanged` attributes from the directory service. When creating a vendor form, add one of these fields to store a time stamp. In the Advanced Search Bar, enter a query for records that meet your time-stamp criteria. For example, use `modifyTimeStamp >= "8/9/2007 4:00:00 PM"` to consider only records modified after 4:00 PM on 8/9/07.

This query is evaluated by the plug-in, which uses it to query the directory server so that it returns only records modified after a specified time.

Caching

The ARDBC LDAP plug-in uses client-side caching. Before a search request is sent to the directory server, AR System checks the cache to determine whether the same request was made before. If an earlier request was cached, the search results are retrieved from the cache to avoid running a new search on the server.

Use the ARDBC LDAP Configuration form to enable caching and to control caching by specifying the maximum size of the cache and the maximum amount of time to keep an item in the cache.

ARDBC LDAP vendor form

If you have problems with your ARDBC LDAP vendor form, consider these tips:

- Any field (except display-only fields) on the vendor form must reference an LDAP attribute that exists in the specified context. For example, if you use MS Active Directories, the `uid` attribute does not exist by default and should not be referenced in your vendor form. If you specify invalid attributes, you might receive unexpected results on your searches.
- If data is not being returned correctly, create a vendor form with only a Request ID and one other field (referring to valid LDAP attributes). Test a search. If it works, continue adding fields until you identify the one that does not work.
  - If any values are NULL, you receive ARERR (100) Entry ID list is empty, and no records are displayed in the client.
  - If more than one record has the same value, you retrieve data only for the first matching entry.
  - For most LDAP servers, `dn` is the attribute of choice for the Request ID. For MS Active Directories, `sAMAccountName` is usually a good choice.

- For optimal performance, set the Directory Page Size field to 1000.
- If you configure the Base DN For Discovery field, the plug-in searches from this Base DN rather than from the root DN. This offers better performance.

AREA LDAP plug-in

The AR System External Authentication (AREA) LDAP plug-in enables you to authenticate AR System users against external LDAP directory services. This section describes how to configure the AREA LDAP plug-in and your AR System server to use LDAP authentication.

After you configure your AR System server, you configure AR System to use external authentication processing. See "Configuring authentication processing" on page 155.

Configuring the AREA LDAP plug-in

To configure the AREA LDAP plug-in, use the AREA LDAP Configuration form in the AR System Administration Console. The settings you specify in the form are saved in the `ar.cfg` or `ar.conf` file. As of release 7.0, AR System supports multiple AREA LDAP configurations.

NOTE

The form is added to your system when you install the plug-in. If you did not install the plug-in during installation of the AR System server, you can install it by rerunning the AR System server installer and selecting the AREA LDAP plug-in installation option. See the *Installation Guide*.

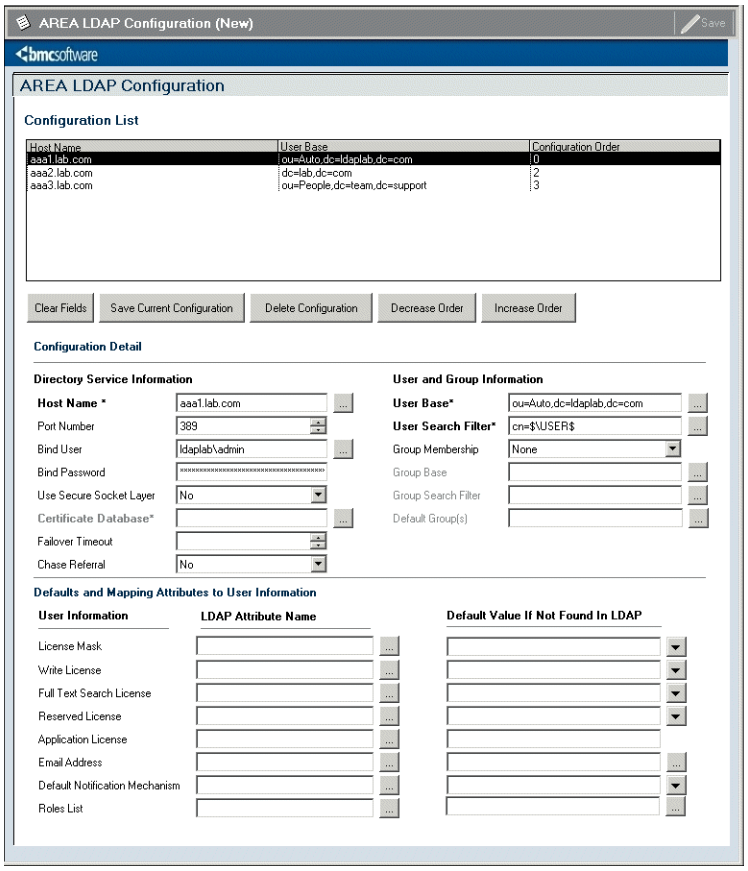
Before configuring the AREA LDAP plug-in, set up user and group information in an LDAP directory service. Then, use the following procedure to enter the settings into the AREA LDAP Configuration form.

▶ To configure settings for the AREA LDAP plug-in

1 In the AR System Administration Console, click System > LDAP > AREA Configuration.

The AREA LDAP Configuration form appears.

Figure 8-2: AREA LDAP configuration form



AREA LDAP Configuration (New)

AREA LDAP Configuration

Configuration List

Host Name	User Base	Configuration Order
aaa1.lab.com	ou=Auto,dc=ldaplab,dc=com	0
aaa2.lab.com	dc=lab,dc=com	2
aaa3.lab.com	ou=People,dc=team,dc=support	3

Configuration Detail

Directory Service Information

Host Name *	aaa1.lab.com
Port Number	389
Bind User	ldaplab\admin
Bind Password	************************************
Use Secure Socket Layer	No
Certificate Database*	
Failover Timeout	
Chase Referral	No

User and Group Information

User Base*	ou=Auto,dc=ldaplab,dc=com
User Search Filter*	cn=$\USER$
Group Membership	None
Group Base	
Group Search Filter	
Default Group(s)	

Defaults and Mapping Attributes to User Information

User Information	LDAP Attribute Name	Default Value If Not Found In LDAP
License Mask		
Write License		
Full Text Search License		
Reserved License		
Application License		
Email Address		
Default Notification Mechanism		
Roles List		

If any AREA LDAP adapters are configured for your AR System server, they are displayed in the Configuration List at the top of the form. When AR System attempts to authenticate a user, it searches each LDAP adapter configuration in the list.

2 In the Configuration List, perform one of these actions:

- To create a configuration, click Clear Fields. All fields in the form are cleared.
- To modify a configuration, select it in the list. The fields in the form are populated with data from that configuration.

3 In the **Directory Service Information** section, fill in (for new configuration) or change (for modified configuration) the values in these fields:

- **Host Name**—Name of one or more servers on which the directory service is hosted**.** You can specify a space-separated list of host names up to 255 characters long. Starting with the first host name in the list, AR System tries to connect to each server until it is successful.
- **Port Number**—Number of the port on which the directory service is listening.
- **Bind User**—Distinguished name for this configuration. The distinguished name is the name for a user account that has read permissions and can search the directory service for user objects.
- **Bind Password**—Password for the distinguished name specified in step n.
- **Use Secure Socket Layer?**—Yes / No toggle field. To specify an SSL connection to the directory service, select **Yes** to enable the Certificate Database field.
- **Certificate Database**—Name of the *directory* containing the certificate database file. To create a certificate database, see "Adding a certificate to a certificate database" on page 401.
- **Failover Timeout**—Number of seconds in which the directory service must respond to the plug-in server before an error is returned. Minimum value is `0` (connection must be made immediately). This value cannot be higher than the value of the `External-Authenticaion-RPC-Timeout` parameter.
- **Chase Referral**—Yes / No toggle field. When the AREA LDAP plug-in sends a request to a directory server, the server might return a referral to the plug-in if some or all of the requested information is stored in another server. Attempting to chase the referral by connecting to the other server can cause authentication problems. By default, referrals are not chased. **Yes** enables automatic referral chasing by the LDAP client. **No** prevents referral chasing.

NOTE

This option is only for Microsoft Active Directory servers. Select **No** for all other directory servers.

IMPORTANT

AR System does not support referrals that use a domain name rather than a host name as a reference. When Active Directory automatically configures referrals (such as when a trust or parent / child domain relationship is created), it uses a domain name in the referral. Therefore, such referrals do not work in AR System even when Chase Referral is set to Yes.

4 In the **User and Group Information** section, fill in or change the values in these fields:

- **User Base**—Base name of the search for users in the directory service (for example, `o=remedy.com`).
- **User Search Filter**—Search criteria for locating user authentication information. You can enter the following keywords in this field. At run time, the keywords are replaced by the values they represent.

  `$\USER$`—Name of the user logging in (for example, `uid=$\USER$`).

  `$\DN$`—Distinguished name of the user logging in.

  `$\AUTHSTRING$`—Value users enter in the Authentication String field when they log in.

  `$\NETWORKADDR$`—IP address of the AR System client accessing the AR System server.
- **Group Membership**—If this user belongs to a group, select Group Container; otherwise, select None. When None is selected, the Group Base, Group Search Filter, and Default Group(s) fields are disabled.
- **Group Base**—Base name of the search for groups in the directory service that includes the user who is logging in (for example, `ou=Groups`).
- AR System performs a subtree search within the group you specify.
- **Group Search Filter**—Search criteria for locating the groups to which the user belongs. For the user's distinguished name, enter the keyword `$\DN$` (for example, `uniqueMember=$\DN$`). At run time, `$\DN$` is replaced with the distinguished name.
- **Default Group(s)**—If the search finds no matching groups, the group specified in this field is used.

5 In the **Defaults and Mapping Attributes to User Information** section, perform these actions:

In the **LDAP Attribute Name** column, enter the corresponding LDAP attribute names for the following AR System fields.

In the **Default Value If Not Found In LDAP** column, select or enter a default value for each field if no value is found in the directory service.

- **License Mask**—Number for the license mask. The license mask specifies whether the AREA plug-in overrides existing information from the User form for write and reserved licenses. It also specifies which license types are overridden by the value returned by the plug-in. Use a number from the following table. An X in a license type column means that the value returned from the plug-in overrides that license in the User form for the specified user.

License mask number	Overridden license types			
	Application	FTS	Reserved	Write
0				
1				X
2		X		
3		X		X
4			X	
5			X	X
6		X	X	
7		X	X	X
8	X			
9	X			X
10	X	X		
11	X	X		X
12	X		X	
13	X		X	X
14	X	X	X	
15	X	X	X	X

- **Write License**—Type of AR System license assigned to the user (Fixed, Read, Floating, or Restricted Read).
- **Full Text Search License**—Type of FTS license assigned to the user.
- **Reserved License**—License type to select for a reserved license.
- **Application License**—Name of the application license granted to the user.
- **Email Address**—Default email address for notifications sent to the user.

- **Default Notification Mechanism**—Notification method used in your environment (none, alert, email, or default).
- **Roles List**—Name of the LDAP attribute that lists the user roles. For example, the `roledn` attribute contains role definitions for some LDAP systems. Add any default roles to the Default Value If Not Found In LDAP field.

6 Click Save Current Configuration.

The system updates the `ar.cfg` or `ar.conf` files with the parameters you specified in this form.

7 (optional) To change the order in which AR System searches the listed configurations when attempting to authenticate a user, do this:

a In the Configuration List, select the appropriate configuration.

b Click one of these buttons:

- **Decrease Order**—Moves the selected configuration down in the authentication attempt order.
- **Increase Order**—Moves the selected configuration up in the authentication attempt order.

NOTE

To make the changes take effect, restart your AR System server.

▶ To delete configurations for the AREA LDAP plug-in

1 In the AR System Administration Console, click System > LDAP > AREA Configuration.

The AREA LDAP Configuration form appears.

2 In the Configuration List, select the configuration to delete.

3 Click Delete Configuration.

The system removes the corresponding parameters from the `ar.cfg` or `ar.conf` files.

NOTE

To make the changes take effect, restart your AR System server.

Configuring AREA LDAP group search

In releases previous to AR System 7.0, external authentication required that every LDAP group to which a user belonged have a matching AR System group. If a user belonged to an LDAP group without a matching AR System group, external authentication failed. Hence, administrators had to create an AR System group for each LDAP group, and AR System searched for groups at only one level in the defined base group. Now, you can *map* LDAP groups to AR System groups and *ignore* excess LDAP groups.

Mapping LDAP groups to AR System groups

This section explains how to map LDAP groups to AR System groups.

NOTE

For maximum benefit, map LDAP groups to AR System groups *and* ignore excess LDAP groups (see "Ignoring excess LDAP groups" on page 152).

▶ To map LDAP groups to AR System groups

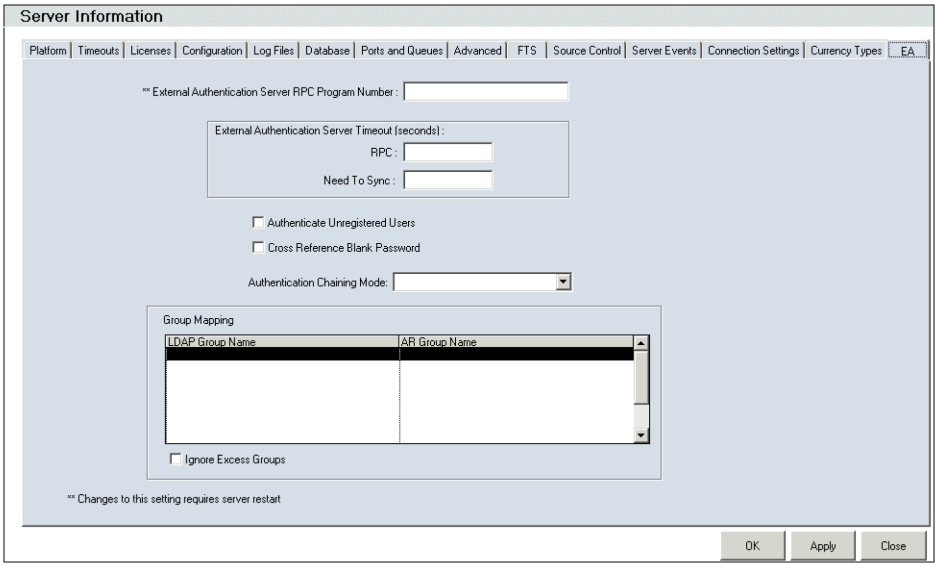
1 Open the AR System Administration: Server Information form, and click the EA tab.

2 Click in the Group Mapping table to add a row, and enter the names of the LDAP and AR System groups to map. Enter only one group name in each column.

NOTE

You can map many LDAP groups to a single AR System group. If you map a single LDAP group to many AR System groups, AR System uses only the first mapping.

Figure 8-3: LDAP Group Mapping table on EA tab



3 Click Apply and OK.

Ignoring excess LDAP groups

Formerly, a user was authenticated only when *each* LDAP group to which the user belonged matched an AR System group. Now, you can configure AR System to authenticate a user when *any single* LDAP group to which the user belongs matches an AR System group. You do this by specifying that AR System ignore excess LDAP groups.

NOTE

For maximum benefit, ignore excess LDAP groups *and* map LDAP groups to AR System groups (see "Mapping LDAP groups to AR System groups" on page 151).

To configure AR System to ignore excess groups

1 Open the AR System Administration: Server Information form, and click the EA tab.

2 In the Group Mapping box, select the Ignore Excess Groups check box.

3 Click Apply and OK.

Configuring AR System servers to use the AREA LDAP plug-in

To configure AR System servers to work with the AREA LDAP plug-in, use the EA (external authentication) tab in the AR System Administration: Server Information form. External authentication (including chaining) works only if you do the following in the EA tab:

- Set RPC to 390695
- Select either Authenticate Unregistered Users or Cross Reference Blank Password or both

For more information, see the *Configuration Guide*.

What's next?

After you configuring your AREA LDAP plug-in and your AR System server, you configure AR System to use external authentication processing. See "Configuring authentication processing" on page 155.

Chapter

9 AR System external authentication

You can use plug-ins and the AR System External Authentication (AREA) API to integrate AR System with external user authentication services. In addition, you can configure AR System to use a combination of internal and external authentication, including OS-based authentication.

The following topics are provided:

- Overview of AREA authentication (page 154)
- Configuring authentication processing (page 155)
- Setting up the AREA hub (page 164)

Overview of AREA authentication

You use the AR System External Authentication (AREA) API to create an AREA server, which mediates between the data source and AR System. (The AREA API is installed with the AR System C API.) The AR System server provides the name, password, and IP address in a remote call to the AREA server, which validates the name and password and then passes the account information back to the AR System server. The AR System server combines the account and user schema information. To implement external authentication, follow these steps:

Step 1 Create a library (.dll or .so) to handle AREA API calls. See these sections:

- "Creating C plug-ins" on page 110
- "Common plug-in C functions and Java methods" on page 123
- "AREA plug-in C API functions" on page 126

Step 2 Link to the AREA library.

Step 3 Install the AREA library on the computer or computers that contain the AR System server.

Step 4 Configure the AR System server to use external authentication. See "Configuring authentication processing" on page 155.

About the AREA LDAP plug-in

AR System includes a sample AREA LDAP plug-in. It is installed during installation of the AR System server if you select the AREA LDAP Directory Service Authentication plug-in option. If you did not select this option during the original installation, you can install the plug-in by rerunning the AR System server installer and selecting the plug-in option.

For information about configuring the plug-in, see "Configuring the AREA LDAP plug-in" on page 145.

Specifying AREA plug-in server settings

For information about configuring your AR System server to work with the AREA LDAP plug-in, see "Configuring AR System servers to use the AREA LDAP plug-in" on page 152.

Configuring authentication processing

To authenticate users, AR System can use internal (User form) authentication, external authentication, or a combination of the two. If you use a combination, you can specify the order in which each type of authentication is attempted.

Specifying when to use internal and external authentication

By default, AR System attempts to authenticate users internally. In all cases, if the user and password match a record in the User form, authentication succeeds. Similarly, in all cases, if the user and password do not match a record in the form, authentication fails.

To use external authentication, select one of the following options in the EA tab in the AR System Administration: Server Information form:

- **Authenticate Unregistered Users**—If a user is not in the User form, AR System tries external authentication. See "Authenticating unregistered users" on page 155.
- **Cross Reference Blank Password**—If a user does not provide a password, AR System tries to cross-reference the user with an external source. See "Cross-referencing blank passwords" on page 156.

IMPORTANT

To use external authentication, you must set the External Authentication Server RPC Program Number field to 390695.

Authenticating unregistered users

When the Authenticate Unregistered Users option is selected, AR System first attempts to find the user in the User form. If the user exists in the User form, AR System attempts authentication through that form. If the user does *not* exist in the User form, AR System attempts authentication through the AREA plug-in.

▶ To authenticate unregistered users

1. Open the AR System Administration: Server Information form, and click the EA tab.
2. In the External Authentication Server RPC Program Number field, enter 390695.
3. Select the Authenticate Unregistered Users check box.
4. Click Apply and OK.

Cross-referencing blank passwords

When the Cross Reference Blank Password option is selected, AR System attempts to authenticate through the User form if the user provides a password. If the user and password match a record in the User form, the user passes authentication. If the user does *not* provide a password, AR System attempts to cross-reference the user with an external system through the AREA plug-in.

▶ To cross-reference blank passwords

1. Open the AR System Administration: Server Information form, and click the EA tab.
2. In the External Authentication Server RPC Program Number field, enter 390695.
3. Select the Cross Reference Blank Password check box.
4. Click Apply and OK.

Specifying authentication chaining mode

You can specify the order in which internal and external authentication methods are attempted by specifying a value for the Authentication Chaining Mode field.

When Authentication Chaining is enabled, all authentication methods in the chain are attempted in the specified order until either authentication succeeds or all the methods in the chain fail.

▶ To set the authentication chaining mode

1. Open the AR System Administration: Server Information form, and click the EA tab.
2. In the External Authentication Server RPC Program Number field, enter 390695.
3. Select Authenticate Unregistered Users, Cross Reference Blank Password, or both.
4. From the Authentication Chaining Mode list, select one of these values:

Mode	Description
`Off`	Disables authentication chaining.
`ARS - AREA`	AR System attempts to authenticate the user by using the User form and then the AREA plug-in.
`AREA - ARS`	AR System attempts to authenticate the user by using the AREA plug-in and then the User form.
`ARS - OS - AREA`	AR System attempts to authenticate the user by using the User form, then Windows or UNIX authentication, and then the AREA plug-in.
`ARS - AREA - OS`	AR System attempts to authenticate the user by using the User form, then the AREA plug-in, and then Windows or UNIX authentication.

NOTE

AR System behaves differently depending on the authentication chaining mode you choose and other external authentication parameters you specify. See "Determining AREA behavior" on page 157.

5 Click Apply and OK.

NOTE

If you use the AREA hub, the authentication chaining mode treats it like a single plug-in, and plug-ins installed in the AREA hub are considered in sequence until a valid response is returned. See "Setting up the AREA hub" on page 164.

Determining AREA behavior

Several factors affect how AR System authenticates users, including these:

- Whether Authenticate Unregistered Users is selected
- Whether Cross Reference Blank Password is selected
- The value of the External Authentication Server RPC Program Number field
- Whether the user exists in the User form and, if so, whether a password exists for the user

The following sections describe AR System authentication behavior for given configurations.

RPC program number is 390695

These tables show AR System authentication behavior for this configuration:

- Authenticate Unregistered Users is selected
- Cross Reference Blank Password is selected
- External Authentication Server RPC Program Number is `390695`

User does not exist in User form

Authentication chaining mode	Authentication behavior
`Off`	■ Authentication is performed using AREA LDAP. User information is retrieved from AREA LDAP.
`ARS - AREA`	■ Authentication is *not* performed using AR System because the user does not exist in the User form. ■ Authentication is performed using AREA LDAP. If successful, user information is retrieved from AREA LDAP.
`AREA - ARS`	■ Authentication is performed using AREA LDAP. If successful user information is retrieved from AREA LDAP. ■ Authentication is *not* performed using AR System because the user does not exist in the User form.

Authentication chaining mode	Authentication behavior
`ARS - OS - AREA`	■ Authentication is *not* performed using AR System because the user does not exist in the User form. ■ Authentication is performed using OS authentication. If successful, user information is retrieved from the OS. ■ If OS authentication fails, user authentication is performed using AREA LDAP. If AREA LDAP authentication is successful, user information is retrieved from AREA LDAP.
`ARS - AREA - OS`	■ Authentication is *not* performed using AR System because the user does not exist in the User form. ■ Authentication is performed using AREA LDAP. If successful, user information is retrieved from AREA LDAP. ■ If AREA LDAP authentication fails, the user is authenticated using OS authentication. If OS authentication is successful, user information is retrieved from the OS.

User exists with no password in User form

Authentication chaining mode	Authentication behavior
`Off`	■ Authentication is performed using AREA LDAP password. User information is retrieved from the User form. ■ Authentication process stops when it fails using AREA LDAP.
`ARS - AREA`	■ Authentication is performed using AREA LDAP password. User information is retrieved from User form. ■ Authentication process stops when it fails using AREA LDAP.
`AREA - ARS`	■ Authentication is performed using AREA LDAP. If successful, user information is retrieved from AREA LDAP. If AREA LDAP configuration does not contain all the information in the form, missing information is retrieved from the `User_Cache`. ■ If AREA LDAP authentication fails, authentication processing stops.
`ARS - OS - AREA`	■ User authentication is performed using AREA LDAP. If successful, user information is retrieved from AR System. ■ If AREA LDAP authentication fails, the user is authenticated using OS authentication. If OS authentication is successful, user information is retrieved from AR System. ■ The user is never authenticated using User form.
`ARS - AREA - OS`	■ User authentication is performed using AREA LDAP. If successful, user information is retrieved from AR System. ■ If AREA LDAP authentication fails, the user is authenticated using OS authentication. If OS authentication is successful, user information is retrieved from AR System. ■ The user is never authenticated using User form.

User exists with password in User form

Authentication chaining mode	Authentication behavior
`Off`	■ Authentication is performed using the AR System User Form. If successful, user information is retrieved from the User form. ■ If User form authentication fails, authentication is *not* attempted using AREA LDAP.
`ARS - AREA`	■ Authentication is performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If User form authentication fails, AREA LDAP authentication is attempted. If AREA LDAP authentication is successful, user information is retrieved from AREA LDAP.
`AREA - ARS`	■ Authentication is performed using AREA LDAP. If successful, user information is retrieved from AREA LDAP. ■ If AREA LDAP authentication fails, authentication is attempted using User form. If User form authentication is successful, user information is retrieved from the User form.
`ARS - OS - AREA`	■ Authentication is performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, OS authentication is attempted. If OS authentication is successful, user information is retrieved from the OS. ■ If OS authentication fails, AREA LDAP authentication is attempted. If AREA LDAP authentication is successful, user information is retrieved from AREA LDAP.
`ARS - AREA - OS`	■ Authentication is performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, AREA LDAP authentication is attempted. If AREA LDAP authentication is successful, user information is retrieved from AREA LDAP. ■ If AREA LDAP authentication fails, OS authentication is attempted. If OS authentication is successful, user information is retrieved from the OS.

RPC program number is 0

These tables show AR System authentication behavior for this configuration:

- Authenticate Unregistered Users is selected
- Cross Reference Blank Password is selected
- External Authentication Server RPC Program Number is `0`

User does not exist in User form

Authentication chaining mode	Authentication behavior
All Authentication chaining modes	■ Authentication is performed using OS authentication. If successful, user information is retrieved from the User form. ■ If OS authentication fails, authentication processing stops.

User exists with no password in User form

Authentication chaining mode	Authentication behavior
All Authentication chaining modes	■ Authentication is performed using OS authentication. If successful, user information is retrieved from the User form. ■ If OS authentication fails, authentication processing stops.

User exists with password in User form

Authentication chaining mode	Authentication behavior
`Off`	■ Authentication is performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, authentication processing stops.
`ARS - AREA`	■ Authentication is performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, OS authentication is attempted. If OS authentication is successful, user information is retrieved from the OS.
`AREA - ARS`	■ Authentication is performed using OS authentication. If successful, user information is retrieved from the OS. ■ If OS authentication fails, User form authentication is attempted. If AR System authentication is successful, user information is retrieved from the User form.

Authentication chaining mode	Authentication behavior
ARS - OS - AREA	■ Authentication is performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, OS authentication is attempted. If OS authentication is successful, user information is retrieved from the OS.
ARS - AREA - OS	■ Authentication is performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, OS authentication is attempted. If OS authentication is successful, user information is retrieved from the OS.

Configuring Atrium SSO integration

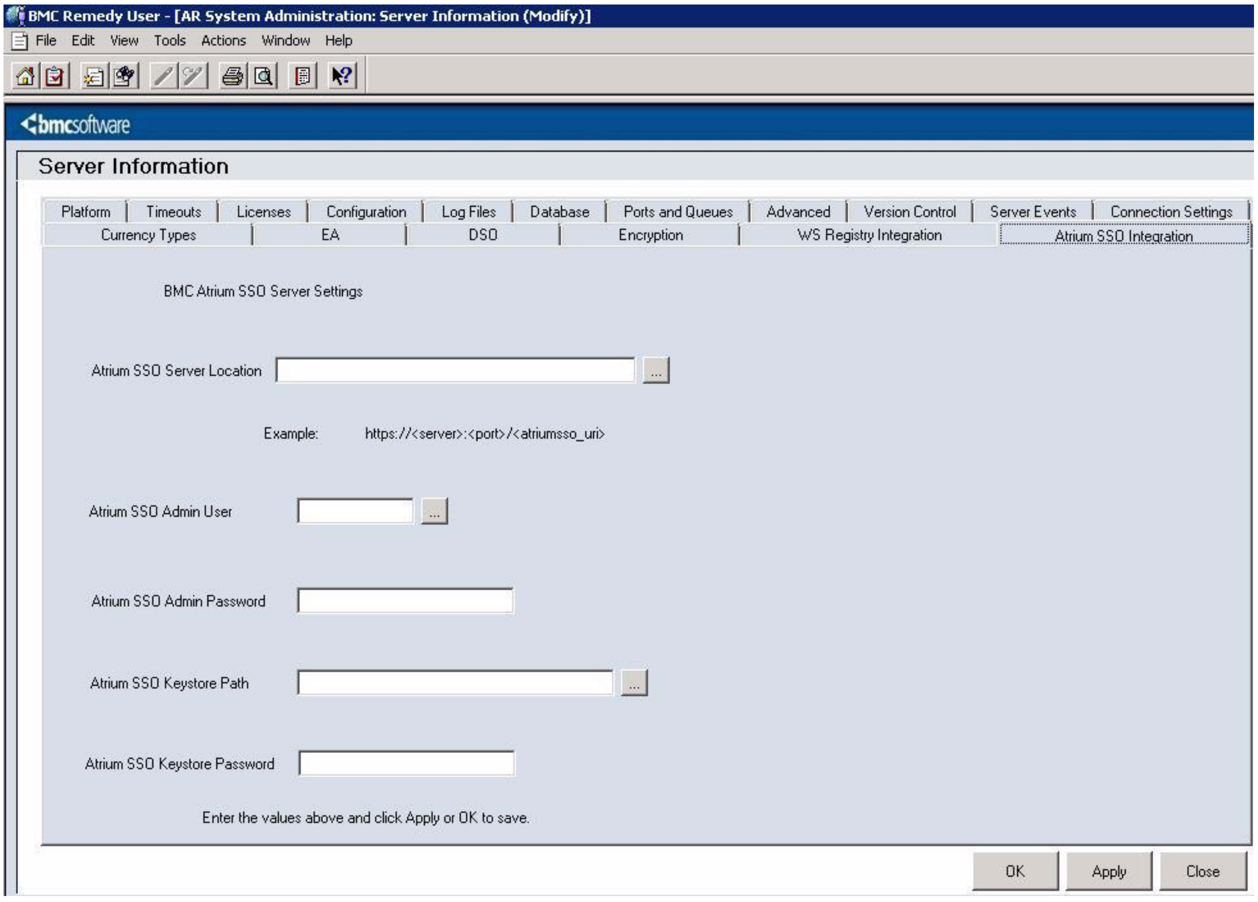
To activate the connection to BMC Atrium SSO, use the Atrium SSO Integration tab of the AR System Administration: Server Information form.

▶ To configure the connection to the BMC Atrium SSO Solution

1 In a browser or BMC Remedy User, open the AR System Administration Console, and click System > General > Server Information.

2 In the AR System Administration: Server Information form, click the Atrium SSO Integration tab.

Figure 9-1: AR System Administration: Server Information form—Atrium SSO Integration tab



BMC Atrium SSO Server Settings

Atrium SSO Server Location

Example: https://<server>:<port>/<atriumsso_uri>

Atrium SSO Admin User

Atrium SSO Admin Password

Atrium SSO Keystore Path

Atrium SSO Keystore Password

Enter the values above and click Apply or OK to save.

3 Enter the values for the following BMC Atrium SSO Server Settings (configuration parameters):

Table 9-1: Atrium SSO Integration tab fields(Sheet 1 of 2)

Field Name	Description
Atrium SSO Server Location	The Atrium SSO Server Location consists of the following information: ■ Host Name—The host name of the computer where Atrium SSO server is configured. If the AR System server and Atrium SSO server are in same domain, enter the machine name or the machine name with domain name. Make sure that the Atrium SSO host name is accessible from the machine where AR System server is installed. If the AR System server and Atrium SSO server are in different domains, a trust relationship between these two domains must be established before configuring Atrium SSO server. ■ Port number—The port on which Atrium SSO server is configured. ■ Protocol—(optional parameter) The default value for this parameter is https. However, this field can also be set to http. For example: `https://<server>:<port>/<AtriumSSO-URI>`
Atrium SSO Admin User	The Atrium SSO administrator name.
Atrium SSO Admin Password	The Atrium SSO administrator password.

Table 9-1: Atrium SSO Integration tab fields(Sheet 2 of 2)

Field Name	Description
Atrium SSO Keystore Path	The keystore file location where the Atrium SSO keystore is saved. This path includes the keystore file name.
Atrium SSO Keystore Password	Password for the keystore.

4 Click Apply.

Manually configuring the mid-tier for Atrium SSO user authentication

For the midtier to communicate with the Atrium SSO server for user authentication, follow the steps given below to manually configure the midtier.

IMPORTANT

If you do *not* select the Configuration of Atirum SSO option during the AR System server installation or during the stand-alone installation of mid-tier, only then perform the steps in this section.

1 Stop the midtier service, if it is already running.

2 Copy all the jar files from the `*WebAgent*\dist\jee\WEB-INF\lib` directory to the `*MidtierInstallDir*\WEB-INF\lib` directory (for example, `C:\Program Files\BMC Software\ARSystem\midtier\WEB-INF\lib`.)

3 Go to the `*MidtierInstallDir*\Web-Inf` directory and open the `web.xml` file in an editor.

4 Uncomment the `<filter>` and `<filter-mapping>` tags for the Atrium SSO filter.

5 Go to the `*MidtierInstallDir*\Web-Inf\classes` directory (for example, `C:\Program Files\BMC Software\ARSystem\midtier\WEB-INF\classes`) and open the `config.properties` file in an editor.

6 Add an attribute in the `config.properties` file. For this, uncomment the DefaultAuthenticator line (`arsystem.authenticator=com.remedy.arsys.session.DefaultAuthenticator`) and add the following line for the Atrium SSO Authenticator:

```
arsystem.authenticator=com.remedy.arsys.sso.AtriumSSOAuthenticator
```

7 Copy the `cacerts` file from the `C:\ProgramFiles\Java\jdk1.6.0_18\jre\lib\security\` directory or from any other location where JDK is located and place it under the `conf` folder in the `Tomcat` directory generally located at `C:\Program Files\Apache Software Foundation\Tomcat6.0\conf\`.

8 Execute the deployer script to deploy the WebAgent. For this, run the following script through command line interface under the `deployer` directory (`webagent\deployer`):

```
java -jar deployer.jar --install --container-type tomcat --atrium-
sso-url AtriumSSOURL<FQDNofAtriumSSOServer>:<port>/atriumsso --
web-app-url MidtierSSOURL<FQDNofMidtierServer>:<port>/arsys --
container-base-dir AppServerHome --admin-name
AtriumServerAdminUsername --admin-pwd AtriumServerAdminPassword --
jvm-truststore "JavaHome\jre\lib\security\cacerts" --jvm-
truststore-password TruststorePassword --truststore
"AppServerHome\conf\cacerts" --truststore-password
TruststorePassword
```

For example:

```
java -jar deployer.jar --install --container-type tomcat --atrium-
sso-url https://ibmc-7ws26bs.adprod.bmc.com:8443/atriumsso --web-
app-url http://ibmc-7ws26bs.adprod.bmc.com:8080/arsys --container-
base-dir  "c:\Program Files\Apache Software Foundation\Tomcat6.0" -
-admin-name amadmin --admin-pwd Let$in09 --jvm-truststore
"c:\Program Files\Java\jdk1.6.0_21\jre\lib\security\cacerts" --
jvm-truststore-password changeit --truststore "c:\Program
Files\Apache Software Foundation\Tomcat6.0\conf\cacerts" --
truststore-password changeit
```

9 Start the midtier service.

NOTE

By default, this plug-in is configured to work with the native plug-in server (C plug-in). You can also use this plug-in directly with the Java plug-in server. For more information on the configuration settings, see “Java plug-in server” on page 116.

Setting up the AREA hub

The AR System plug-in server supports only one AREA plug-in directly. You can, however, add a single AREA Hub plug-in to the plug-in server and then add multiple AREA plug-ins to the AREA Hub plug-in. Plug-ins you add to the AREA Hub are referred to as Hub-plug-ins. The AREA Hub propagates the calls it receives from the Hub-plug-ins to the Plug-in server.

The AREA Hub loads the Hub-plug-ins in the order in which they appear in the `ar.cfg` or `ar.conf` file. That is, the first entry the AREA Hub finds in the `ar.cfg` file is the first plug-in loaded, the second entry the second, and so on.

NOTE

You do not need to configure the AREA Hub manually to use multiple AREA LDAP plug-ins. See “Configuring the AREA LDAP plug-in” on page 145.

▶ To set up the AREA Hub plug-in

1 Create the following entry for the AREA Hub in the `ar.cfg` file:

```
plug-in: areahub.dll
```

NOTE
Make sure this is the only entry for an AREA plug-in in your `ar.cfg` file.

2 Create an entry for the first AREA plug-in as follows:

```
AREA-Hub-Plugin: my_area_plug-in.dll
```

3 If necessary, create entries for subsequent AREA plug-ins as follows:

```
AREA-Hub-Plugin: my_area_plug-in_1.dll
AREA-Hub-Plugin: my_area_plug-in_2.dll
AREA-Hub-Plugin: my_area_plug-in_3.dll
```

4 Restart the AR System plug-in server.

NOTE
For information about restarting the plug-in server, see "Restarting the plug-in server using the Set Server Info command" on page 118.

Chapter 10 Data visualization fields

This section provides an overview of data visualization fields and their use in enabling graphical elements in AR System forms.

The following topics are provided:

- Overview (page 168)
- Services provided to the data visualization modules on BMC Remedy Mid Tier (page 168)
- Services provided on clients (page 170)
- Using Java classes (page 171)
- Working with native libraries (page 172)
- Storing shared library files on the mid tier (page 172)
- Creating data visualization fields (page 174)
- Configuring right-to-left format in a data visualization field (DVF) (page 182)

Overview

AR System provides a seamless integration of web content through the data visualization field (DVF). This integration deploys the web content generation module (the server-side component that generates web content) through the data visualization module (DVM).

Because the DVM is a server-side component, it needs the BMC Remedy Mid Tier as a hosting environment. The mid tier must be running if you want to view the web content while authoring in BMC Remedy Developer Studio.

For example, a flashboard is implemented as a DVM. To see the flashboard during authoring time, the mid tier must be running, and the AR System server must be configured so that it can obtain default web content from the mid tier (set the Default Web Path setting on the AR System Administration: Server Information form).

The DVF provides the following benefits:

- Data visualization module developers need to write only the image generation and user interface code.
- Authentication is handled automatically.
- Authorization for the definitions is handled automatically.
- Ease of deployment and version control. Modules are automatically deployed on all mid tier systems that get requests for the data visualization.
- Data visualization definition objects smoothly integrate into the AR System import/export model.
- Client-side workflow integration is supported. (The mid tier can generate code for such integrations when users click hotspots.)
- By implementing the PluginContext interface and appropriate data model pluggability, the code for a module can be reused in environments that do not include BMC Remedy Mid Tier.

Services provided to the data visualization modules on BMC Remedy Mid Tier

The data visualization feature includes the following services to ease integration with AR System.

- **Authentication services**—Supplied by BMC Remedy Mid Tier.
- **Authorization services**—Supplies meta-information needed to generate the graphical elements. This information is retrieved only if a user has permission to view the graphical elements. The meta-information is stored in a special form.

- **Deployment services**—The module code is stored as a JAR file attachment on the AR System server. The module container automatically downloads this code from the AR System server and uses it to serve up requests. This makes installation and deployment of the module code a simple data import operation into a form on the AR System server.
- **Signaling (event) services**—Signals (events) are sent on the client side. For the module writer to use signals, the module needs to generate HTML content in response to requests. These responses can include an HTML script tag that contains event dispatching JavaScript code that is generated by the module container. This script snippet can be obtained at the mid tier by calling the `PageService.getEventInfrastructureCode()` method. The PageService also returns the name of the `Javascript eventDispatcher` object in the `PageService.getEventDispatcherName()` method. It can be used to send and receive events on the client side. This name might change in portlet environments to prevent name clashes.

  The following signals are available:

  - **Signals from module client side to module mid tier**—Generates internal code for sending signals to the mid tier. Module code must be written to call the `EventDispatcher.sendEventToMidTier(eventtype, eventData)` JavaScript object method in the HTML response that is generated by the module in response to a request, which returns a string value.
  - **Signals from module client-side to parent AR System form**—Generates the internal code that sends signals from the module in the main form. Module code must be written to call the `EventDispatcher.sendEventToParent (eventType, eventData)` JavaScript object method in the HTML response that is generated by the module.
  - **Signals from AR System form workflow to module**—Generates the code to receive events from the main form in the module. Module code must be written to register a JavaScript function with the EventDispatcher java script object on the client side by calling `EventDispatcher.setEventHandler(object, functionRefence)`. The *object* parameter is the name of a variable that refers to the object that has the `functionReference` as one of its members. For functions that are not members of an object, null can be passed in for the object parameter.
  - **Drill down from client module to an AR System form**—Generates code to perform the drill-down into an AR System form without being dependent on the module HTML being present in an AR System form. This allows the module code to generate HTML that can be just fragments in a portlet environment, while still providing the drill-down capability.

    The function call is `EventDispatcher.drillDownToForm(server, form, view, qualification)`.

- **Configuration services:**
  - Surfaces configuration properties of the module into the BMC Remedy Mid Tier configuration page.
  - Stores the configuration properties and sends them to the module when it is initialized.
- **Definition storage and retrieval services**—Definitions in AR System are retrieved from the AR System server and sent to the module when it is asked to handle a request.
- **Caching services**—The module container can be requested to cache objects created and repeatedly used by the module.
- **Locale services**—Retrieves messages for a specific locale for a supplied key. Numbers, dates, and times are formatted based on the locale.
- **Page generation services**—URLs are generated for various types of hotspots used in signaling for this module.
- **Session services**—Enables module objects to be stored in the user session so that they are available throughout the lifetime of the session.

Services provided on clients

The main services provided on the clients are the signaling services. For sending signals to the data visualization module from an AR System form, remember the following guidelines:

- Workflow must be written to call the `PERFORM-ACTION-SEND-EVENT` Run Process action, specifying the target window ID as a field by specifying the ID as `F<`*`fieldID`*`>`.
- The `$EVENTTYPE$` and `$EVENTDATA$` keywords are used to pass the event type and data separately, instead of using only the `$EVENTTYPE$` keyword.
- The Send Event Run Process uses an additional argument to represent the event data (`$EVENTDATA$`). This argument is a string that must be enclosed in double quotation marks if it contains spaces (with the AR System standard of escaping of double quotation marks being two double quotation marks together, representing a single double quotation mark character.)
- To send signals to the mid tier, call the `EventDispatcher.sendEventToMidTier(evttype, evtData)` through a JavaScript call.
- To send signals to the parent form, call `EventDispatcher.sendEventToParent(evtType, evtData)`. This results in an event being generated on the AR System form containing the data visualization field. Workflow can be added to act on an event with the run if condition being defined on the `$EVENTTYPE$/$EVENTDATA$/$EVENTWINSRCID$` keywords. The `$EVENTTYPE$` and `$EVENTDATA$` keywords are set to the values passed in the call. The `EVENTSRCWINID` is set to `F<`*`moduleFieldID`*`>`.

- To drill down to another form, call `EventDispatcher.drillDownToform(server, form, view, qualification)` on the client side opens the form from the server (if current credentials are valid for that server) in Modify mode, with the entries matching the qualification displayed in the results list. Use the signaling to parent form mechanism with appropriate workflow on the main form to drill-down into forms and have them open up in modes other than modify.

Using Java classes

Data visualization modules are packaged and deployed as JAR files. The following Java classes and interfaces are provided for data visualization developers. The Javadocs contain more information about these classes.

NOTE

When searching for a class name, remember that each class is in the `com.remedy.arsys.plugincontainer` package. So, the `Plugin` class is named `com.remedy.arsys.plugincontainer.Plugin`.

Data visualization developers can implement these interfaces:

- **Plugin**—Interface that must be implemented by data visualization developers.
- **DefinitionFactory**—Interface that can optionally be implemented by data visualization developers to parse their definition data into Java objects. These Java objects must imported into the Definition or the CacheableDefinition interfaces described in the following list.
- **Definition**—Marker interface implemented by the modules' definition objects.
- **CacheableDefinition**—Marker interface to use if the module needs the container to cache the definition objects. The cached objects are returned when the module requests the DefinitionSevice to get the definition. For more information, see the cache service.

Other interfaces include:

- **ARPluginContextEX**—Extends the PluginContext class to provide AR System specific services. To use this class in a plug-in, you must typecast objects to `com.bmc.arsys.api.ARServerUser`. (In version 7.0.01, the ARPluginContext is used, and it returned `com.remedy.arsys.api.ARServerUser` object in its functions. For more information, see the 7.6.04 AR System API Javadocs.)
- **AuthenticationException**—Extends the Exception class.
- **CacheableDefinition**—Uses the caching services of the plug-in container.
- **DefaultDefinition**—Default implementation of the Definition class.
- **Definition**—Acts as a marker interface for data visualization definition.
- **DefinitionService**—Provides the definition service for the data visualization container.

- **LocaleService**—Provides services such as getting appropriate strings for the locale to enable the module writer to localize the plug-in. For example, you can format dates, time, and numbers for locale conventions.
- **ARLocaleServiceEX**—As a subset of LocaleService, provides a simpler way to access localized messages from the Message Catalog.
- **PluginContext**—Provides the context that can be used to make API calls.
- **NoPermissionException**—Occurs when the data visualization tries to retrieve a definition for a user who does not have authorization to access the definition.
- **PageService**—Generates URLs that provide a reference back to the data visualization.

NOTE

When you update to new APIs, make sure you make the appropriate changes (as described in the Javadocs).

Working with native libraries

When the Java virtual machine (JVM™) executes a Java application that requires a native library through the Java native interface (JNI™), the JVM must load the native library exactly once. This library (exposed to the Java layer by JNI) remains available as long as the JVM process is active. If a *different* Java application using the same native library executes on the same JVM process, it tries to load the native library again. Because the library is already loaded, an exception is generated.

The JNI developer must make sure this native library is loaded exactly once and must handle the exception if the same library is asked to be loaded again.

NOTE

The Java API does not use the JNI layer to connect to the AR System 7.5 (or above) server. However, the JNI layer is still used to connect to pre 7.5 versions of the server.

Storing shared library files on the mid tier

For a plug-in (such as CI Viewer, which is packaged with BMC Atrium CMDB) to work, the plug-in's installer copies the required native libraries to the mid-tier installation folder. When mid tier is upgraded or re-installed, these shared library files are deleted, and you must re-copy the required native libraries to the mid-tier installation folder.

To avoid re-copying the required native libraries, store each library file as an entry in the **Data Visualization System Files form** in AR System. When the mid-tier process starts, a service looks for this form on all of the servers registered as Data Visualization servers, and workflow copies the library files to the mid tier.

If the file is already available in the mid-tier installation directory and the modified date matches the last-modified date of the entry in the Data Visualization System Files form, the file is not re-copied.

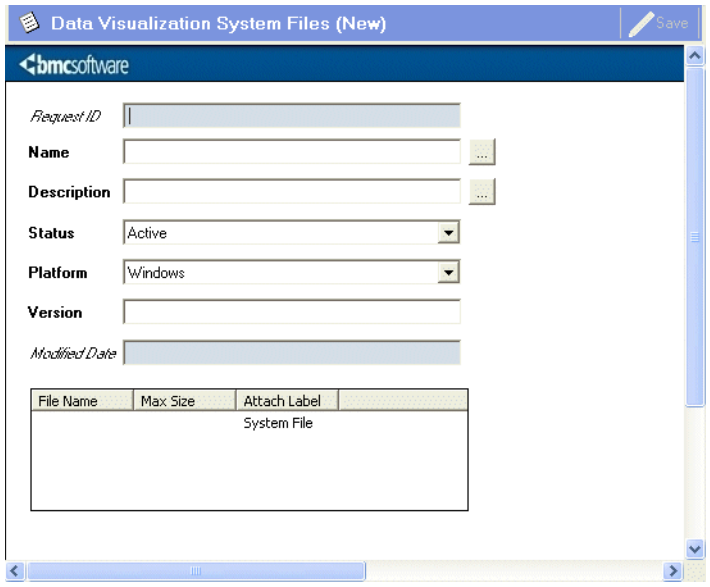
NOTE

The Data Visualization forms are installed with the AR System server. For the Data Visualization System Files form to function, the AR System and mid-tier servers must be version 7.1.00 or later.

▶ To store native library files for plug-ins

1 Open the Data Visualization System Files form.

Figure 10-1: Data Visualization System Files form



Data Visualization System Files (New)

Request ID	
Name	
Description	
Status	Active
Platform	Windows
Version	
Modified Date	

File Name	Max Size	Attach Label
		System File

2 In the Name field, enter the file's name.

IMPORTANT

This name must exactly match the name of the file that is attached at the bottom of the form. If the name does not match, the data visualization field will show an error message when the application within the field is run.

3 Enter a description.

4 Enter the status.

The options are Active or Inactive. If the status is Inactive, the file is not copied during an upgrade or re-installation.

5 Select the platform:
   - Windows
   - Solaris
   - Linux
   - AIX
   - HP UX
   - All—This option is used for `.jar` files, which are used on multiple platforms.

   The mid tier copies only those files that match the platform you select, except for the All option, which copies `.jar` files.

   NOTE
   Entries are indexed by the Name and Platform fields.

6 In the Version field, enter the version number of the binary file that you are saving.

   Use the following format:

   `majorVersion.minorVersion.subminorVersion.patch`

   For example, for version 2.0.01, patch 002, enter:

   ```
   2.0.01.002
   ```

   You can omit the trailing numbers. For example, for version 2.1.00.000, you can enter:

   ```
   2.1
   ```

   If the same file is available on a different server with a new version, AR System matches the version string. Then, it uses the latest version.

7 Add the file to the attachment field.

8 Click Save.

Creating data visualization fields

Use the following process to create a data visualization field (DVF).

Step 1 Create a module on the mid tier. (See page 175.)

Step 2 Register the module. (See page 178.)

Step 3 Deploy a custom data visualization module (DVM). (See page 180.)

Step 4 Add a DVF to a form. (See page 181.)

NOTE
If you are adding a flashboard to a DVF, go directly to step 4.

Creating data visualization modules on the mid tier

Use the following procedure to create DVMs.

For a more detailed explanation and examples, see “The Hook Framework for the Data Visualization Field” Knowledge Base article at `http://www.bmc.com/support`. Search with the keywords “data visualization.”

IMPORTANT

When writing the module, be sure to generate output that is compliant with Section 508 if your users specify anything other than default in the Accessible Mode field of the User Preference form.

▶ To create a data visualization module

1 Save your code to a `.java` file with a name that matches the Class file name.

Following the example in “Example: HelloWorld plug-in,” which follows this procedure, your resulting file should be `HelloWorldPlugin.java`.

2 Compile the code using `javac`.

Make sure to reference the `GraphPlugin.jar` file from the mid tier’s `/WEB-INF/lib` directory and a `.jar` file that contains the Oracle Java `HttpServletResponse` class in your `-classpath` parameter for `javac`. `HttpServletResponse` is in `j2ee.jar` if you use a OracleOne web server; it is in `servlet.jar` if you use ServletExec. For example:

```
javac -classpath /yourDirStructure/j2ee.jar:/yourDirStructure/
GraphPlugin.jar HelloWorldPlugin.java
```

3 Add the compiled `.class` file to a `.jar` file. For example:

```
jar -cvf DataVisHelloWorld.jar HelloWorldPlugin.class
```

If you use a package structure, enter this:

```
jar -cvf DataVisHellowWorld.jar topLevelDir
```

If you use package structures, remember these guidelines:

- If you chose to use a Java package reference, add this to the sample code, for example, `package com.mycompany.plugin`.
- When running `javac` and `jar`, your `.java` or `.class` file must exist in a directory structure that matches the package declaration, and you should be in the directory immediately above the top of the package structure when running `javac` and `jar`.

When running `javac`, `java`, and `jar` commands, remember that these Java tools look “down” a directory structure. To see files in directories above your current directory, use the `../` reference; you cannot simply use an absolute file path reference.

To create data visualization modules, you can use the following examples.

Example: HelloWorld plug-in

This example sends back HTML that displays `Hello World` on the client.

```
public class HelloWorldPlugin implements Plugin {
    public void init(PluginConfig config) {
    }
    public void processRequest(PluginContext pc) throws
IOException, NoPermissionException    {
        HttpServletResponse response = pc.getResponse();
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter writer = response.getWriter();
        writer.println("<html><head><title>Hello World Plugin
Example</title></head>");
        writer.println("<body><h1>Hello World</h1></body></
html>");
    }
    public String handleEvent(PluginContext pc, String
eventType,String eventData) throws IOException,
NoPermissionException {
         return "alert(\"Got event data in Midtier as " +
eventData + "\");";
    }
    public DefinitionFactory getDefinitionFactory() {
         return null;
    }
    public void cleanup() {}
}
```

Example: event handling code

This example shows event handling code added to the generated HTML in the head tag.

```
PageService ps = pc.getPageService();
HttpServletResponse response = pc.getResponse();
PrintWriter writer = response.getWriter();
writer.println("<html><head>");
writer.println(ps.getEventInfrastructureCode());
writer.println("<script>function sendMidTierEvent() {");
writer.println("var result = "+ps.getEventDispatcherName()+
".sendEventToMidTier(\"ClickEvent\",\"Title\");");
writer.println("eval(result);");
writer.println("};</script>");
writer.println("</head>");
writer.println("<html><head><title>Hello World Plugin Example</
title></head>");
writer.println("<body><h1 onclick=sendMidTierEvent()>Hello
World</h1></body></html>");
```

Example: definition factory

This is an example of using the definition factory.

```
public void processRequest(PluginContext pc) throws IOException,
NoPermissionException {
    HttpServletRequest req = pc.getRequest();
    String defName = req.getParameter("name");
    // get our definition object from the definition service.
    MyDefObject def = (MyDefObject)
pc.getDefinitionService().getDefinition(name);
    ….. generate the html using def ….
}
private DefinitionFactory myDefFactory;
public void init(PluginConfig config) {
   // create the definition factory and stash it in a member
variable for future use.
    myDefFactory=new MyDefObjFactory();
}
public DefinitionFactory getDefinitionFactory() {
    // return the stashed definition when the definition service
asks for it.
    return myDefFactory;
}
private class MyDefObjectFactory implements DefinitionFactory {
    // This method is called by the AR Plugin container if the
definition is stored
   //  in the SimpleDefinition field in the Data Visualization
Definition form
    public Definition createFromString(PluginContext pc, String
defName, String defAsString) throws IOException {
        return new MyDefObject(defName, defAsString);
    }
    // This method is called by the AR Plugin container if the
definition is stored
   //  in the ComplexDefinition field in the Data Visualization
Definition form
    public Definition createFromStream(PluginContext pc, Strign
defName, InputStream defAsStream) throws IOException {
        return new MyDefObject(defName, defAsStream);
    }
}
// This class implements CacheableDefinition so that the
Definition Service provided by
// the AR Plugin container caches this object till it sees any
modifications in the entry for
// this definition in the Data Visualization Definition form.
private class MyDefObject implements CacheableDefinition {
    private MyDefObject(String defName, String defString) {
        … initialize the def object with the string …

    private MyDefObject(String defName, InputStream is) {
        … initialize the def object using the stream …
```

Example: using the locale service to format dates

This snippet is an example of using the locale service for formatting dates and also formatting the AR System `com.remedy.arsys.Value` objects.

```
LocaleService ls = pluginContext.getLocaleService();
// format current date as per current locale and user
preferences
String dateValue = ls.formatDate(new Date());
ARLocaleService als = (ARLocaleService)ls;
Value val = getValueFromServer(server, form, fieldId);
String formattedStr = als.formatValue(val, server,
form,view,fieldId);
```

Example: using the locale service to get a localized string

This snippet is an example of using of the locale service to get a localized string.

```
LocaleService ls = pluginContext.getLocaleService();
String defName=pluginContext.getRequest().getParameter("name");
int TITLE_ID=100;
LocalizedStringID localizedTitle = new
LocalizedStringID(defName, TITLE_ID);
LocalizedStringID[] retrieveIDs={localizedTitle};
String[] localizedStrings = ls.getLocalizedStrings(retrieveIDs);
String localizedTitle = localizedStrings[0];
```

Registering data visualization modules

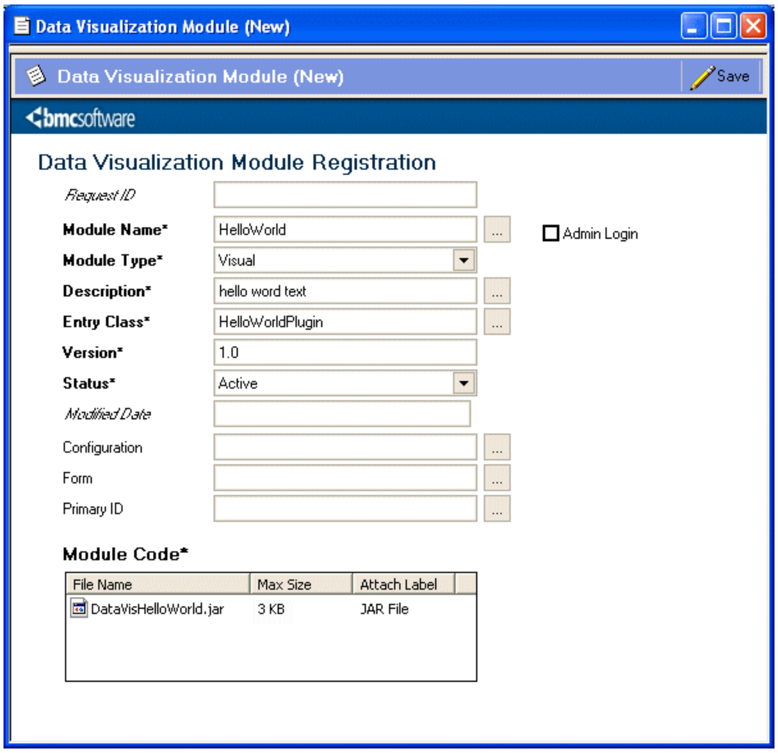
DVMs must registered on an AR System server to be used by AR System forms on the server.

▶ To register a data visualization module

1 In BMC Remedy User, open the Data Visualization Module form in New mode.

This form tells the mid tier server that the plug-in exists.

Figure 10-2: Data Visualization Module form



Data Visualization Module (New)

Data Visualization Module Registration

Request ID	
Module Name*	HelloWorld
Module Type*	Visual
Description*	hello word text
Entry Class*	HelloWorldPlugin
Version*	1.0
Status*	Active
Modified Date	
Configuration	
Form	
Primary ID	

☐ Admin Login

Module Code*

File Name	Max Size	Attach Label
DataVisHelloWorld.jar	3 KB	JAR File

2 In the Module Name field, enter the name of the module definition (the plug-in's name).

3 In the Module Type field, select Visual.

The Data option is reserved for future use.

4 In the Description field, enter the description of the of the data visualization definition.

5 In the Entry Class field, enter the entry class, for example, `com.remedy.arsys.mymodule.Mymodule`.

The value must be the class file you created with `javac`. If you compiled using a package structure, the value for this field must be the complete class name, for example, `com.mycompany.plugin.HelloWorldPlugin`.

6 In the Version field, enter the version of the module.

You can enter any number according to your own version numbering scheme.

If multiple versions of a DVMs are registered, the version with the latest version number is used.

7 In the Status field, select Active or Inactive to define the status of the module.

8 In the Module Code field, attach the JAR file.

9 Click Save.

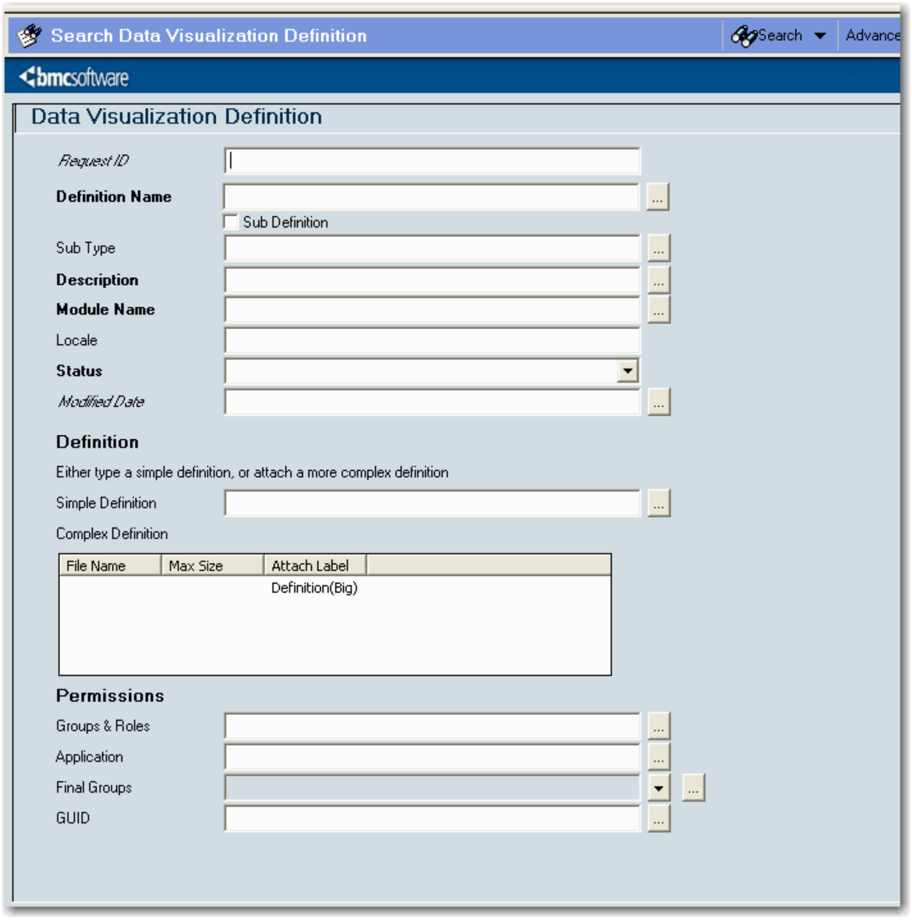
A custom DVM must be deployed to a BMC Remedy Mid Tier system to be used by AR System forms that are accessed from a browser.

▶ To deploy a custom data visualization module

1. Open the BMC Remedy Mid Tier Configuration Tool.
2. In the Module server(s) field on the General Settings tab, enter the names of the AR System servers that contain the modules.
3. In BMC Remedy User, open the Data Visualization Definition form in New mode.

   This form defines the instance of your plug-in.

Figure 10-3: Data Visualization Definition form



Data Visualization Definition

Request ID

Definition Name

Sub Definition

Sub Type

Description

Module Name

Locale

Status

Modified Date

Definition

Either type a simple definition, or attach a more complex definition

Simple Definition

Complex Definition

File Name	Max Size	Attach Label
		Definition(Big)

Permissions

Groups & Roles

Application

Final Groups

GUID

4. In the Definition Name and Description fields, enter the name of the description for the data visualization to deploy.

   The Definition Name is what appears in the data visualization form.
5. In the Module Name field, enter the name of the module name from the Data Visualization Module Registration form. (These names *must* match.)
6. In the Complex Definition table at the bottom of the form, add the JAR file for the data visualization module.

7 Click Save.

After you deploy the module, add the field to the form, as described in "To add a data visualization field to a form."

8 To enable users to open forms with data visualization fields, add the Default Web Path to your server's configuration (go to the Advanced tab of the AR System Administration: Server Information form).

▶ To add a data visualization field to a form

1 Make sure that the BMC Remedy Mid Tier is running so that you can view the web content as you are creating the field.

2 In BMC Remedy Developer Studio, open the appropriate form.

3 Right-click the form, and choose Create a New Field > Data Visualization.

The new data visualization field appears on the form.

4 Select the field.

5 In the Properties tab, click these properties, and select the appropriate value from the drop-down list:

- **Module Type**—The module type for the data visualization, which is defined in the Module Registration form:
  - Flashboard
  - Report
  - Visualizer
- **Server**—The AR System server that contains the data visualization module.
- **Definition Name**—The definition name for the data visualization module.

For more information about field properties, see the *Form and Application Objects Guide*.

6 Select the Custom Properties property, and click its ellipsis button.

7 In the Custom Properties dialog box, enter a customizable parameter.

For example, if you are adding a flashboard to the data visualization field, enter a parameter and value using the following format:

```
parameter=value
```

For more information, see the *BMC Remedy Flashboards Guide*.

8 Right-click the form, and choose Save.

Configuring right-to-left format in a data visualization field (DVF)

A DVF can be a part of more than one view, and because text direction is associated with a view, the text direction for a DVF is not known until the DVF is initialized. This initialization takes place when the browser renders the view of the AR System form. At that time, a request with text direction is sent to the DVF.

If you create a DVF and you want to create different contents based on text direction, extract the text direction from the `PluginContext`. (Because the DVF cannot determine your intent regarding text direction, you must add the text direction to your HTML content for the DVF [for example, by using the `<html dir='rtl'>` option].)

To extract the text direction, use the `com.remedy.arsys.plugincontainer.ARLocaleServiceProperties` interface through the `public String getProperty(String key, String defvalue);` method.

Following is a simple example for extracting the text direction from the `PluginContext` object given `com.remedy.arsys.plugincontainer.PluginContext pc;`

```
ARLocaleServiceProperties p =
(ARLocaleServiceProperties)pc.getLocaleService();
boolean isRTL = false;
String val = p.getProperty("rtl", "false");
if (val.equalsIgnoreCase("true"))
  isRTL = true;
```

Chapter

11 Vendor forms

Vendor forms are AR System objects that let you view and process external data using AR System processes and workflow. This section discusses vendor forms and how to configure your system to use them.

The following topics are provided:

- About vendor forms (page 184)
- Creating vendor forms (page 184)

About vendor forms

Vendor forms allow AR System to present data from external sources as entries in an AR System form. When you create a vendor form, you can request a list of candidate forms or fields (preferred method) or you can enter the information yourself.

Vendor forms require you to have an ARDBC plug-in installed and configured. The ARDBC plug-in and the plug-in server handle data exchange between AR System and the external data source. The AR System server maps the external data to fields in the vendor form, and the form displays the data. See “ARDBC plug-ins” on page 127.

You can use vendor forms to do the following tasks:

- Implement workflow on creation and modification of external data.
- Execute escalations on external data.
- Access external data to populate search style character menus or table fields.

The vendor form can be manipulated as a regular form type with the following exceptions:

- You can add only Required and Optional fields that correspond to actual columns in the data source. In addition, you can add a Display Only field only when the column name does not correspond to a column in the data source.
- Full Text Search (FTS) operations are not available on vendor forms.

Creating vendor forms

You can create a vendor form after you have built and installed your ARDBC plug-in, and configured your server to recognize it. For information about configuring your server to recognize a plug-in, see the *Configuration Guide*.

NOTE
Creating a vendor form for an ARDBC LDAP plug-in is a special case. See “Creating a vendor form to represent a collection of LDAP objects” on page 140.

▶ To create a vendor form using an ARDBC plug-in

1 In BMC Remedy Developer Studio, choose File > New > Vendor Form.

The New Vendor Form Wizard appears.

2 Select the server on which you want to create the vendor form, and click Next.

3 Select the ARDBC plug-in to use in the list of Available Vendor Names, and click Next.

4 Choose a table from the list of Available Vendor Tables and click Next.

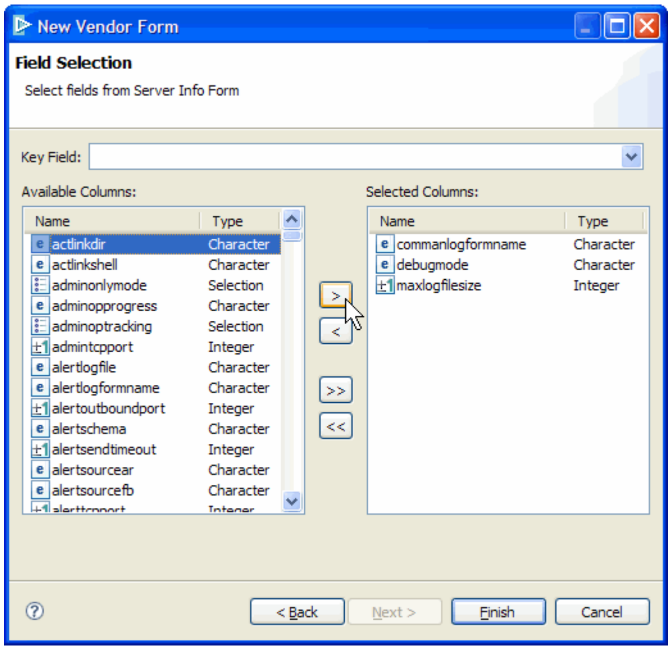
Alternatively, type a table name in the Table field, click Validate, then click Next.

5 (optional) On the Field Selection page, choose a key column in the Key Field list box.

The key column uniquely identifies the entries in your vendor form. Key values are mapped to the Request ID field on the Vendor Form.

6 In the Available Columns list on the Field Selection page, select columns to access in AR System. Use the arrow buttons to move them to the Selected Columns list.

Figure 11-1: New Vendor Form Wizard, Field Selection page



Available Columns:

Name	Type
actlinkdir	Character
actlinkshell	Character
adminonlymode	Selection
adminopprogress	Character
adminoptracking	Selection
admintcpport	Integer
alertlogfile	Character
alertlogformname	Character
alertoutboundport	Integer
alertschema	Character
alertsendtimeout	Integer
alertsourcear	Character
alertsourcefb	Character
alerttcpport	Integer

Selected Columns:

Name	Type
commanlogformname	Character
debugmode	Character
maxlogfilesize	Integer

7 Click Finish to create the vendor form.

8 Use Developer Studio to edit the new form, then click File > Save.

NOTE

For information on creating a join form using a vendor form, see the "Requirements for creating a join form using a vendor form" section in the *Form and Application Objects Guide*.

Chapter

12 View forms

View forms enable AR System to point to and access data in relational database tables outside AR System. The table can be located on the same database instance or in any other database accessible from the current AR System database.

The following topics are provided:

- About view forms (page 188)
- Creating and modifying view forms (page 192)
- Setting up a remote database for view forms (page 194)

About view forms

Because view forms access data outside of AR System, the developer must understand the database data types and must have access to the external database table containing the data. This section describes the requirements for using view forms and the data types supported for each database.

Database requirements for view forms

Before creating a view form, identify the database table to use and verify that the following requirements are met:

- The database table must reside on, or be accessible to, the database that AR System is using.
- The `ARAdmin` user must have read and write access privileges on the database table.
- The database table must have a column (field) that enforces non-null and unique values. This column acts as the Request ID. If the administrator chooses a column that is not unique or that allows nulls, data corruption might occur. The Request ID field must be an integer or character field with 6–15 characters. Otherwise, the Key Field list is empty, and you cannot create the view form. If the administrator chooses a character column for the Request ID, then the field length must be the same as the column length.
- You can use a view form to access BLOBs on a remote database, but not CLOBs.
- Long columns (that is, `text` or `clob`) must allow null values.

There are additional configuration requirements if the table to be used with the view form is on a remote database. See "Setting up a remote database for view forms" on page 194.

Special considerations for view forms with DB2

For DB2® databases, you can connect only to database tables in the same DB2 database as the AR System database.

To use view forms on a DB2 database, you must add the database's user name to the `ar.conf` (`ar.cfg`) file by using the `Db-user` option. For example, if the DB2 administrator's name is `db2admin`, add the following line to the `ar.conf` (`ar.cfg`) file:

```
Db-user: db2admin
```

For more information about the `ar.conf` (`ar.cfg`) file, see the *Configuration Guide*.

AR System requirements for view forms

A view form can be manipulated as a regular form type with these exceptions:

- You can add only Required and Optional fields that correspond to actual columns in the data source. You can add a Display Only field only when the column name does not correspond to a column in the data source.
- After you attach an AR System field to a column in the database table, you cannot reattach the field to a different column, but you can change other field properties.
- Status history, diary, currency, and attachment fields are not supported on view forms.
- You cannot change the type of a text field or change the length of any field after initial creation.
- BCE dates are not supported in date fields in a view form.

Field properties for fields on view forms

Field properties on view forms are the same as the field properties of a regular form, with the following exceptions:

- The View Information category includes the following properties:
  - **Table**—Indicates the link to the external database table. This field cannot be modified.
  - **Column**—Displays the column name on which the field was created. This field can be modified, but it must be 254 characters or less.
- If the column name does not represent a column in the data source, the field must be display-only.

  For example, assume you add a character field to a view form. The Column property shows the column name as `Character Field`, which does not exist in the data source. To save the form, you must change the Column property to match a column in the data source, or set the Entry Mode property to Display.
- If the column name represents a column in the data source, the field cannot be display-only.
- You cannot change the data type of a character field on a view form. You can decrease the input length of a character field, but this action does not alter the corresponding column in the database. The input length should never be increased beyond its initial value.

Database data types for view forms

The following sections list the data types supported by each database for AR System field types in view forms. BMC Remedy Developer Studio automatically maps the field types to corresponding data types.

Beginning with release 7.6.02, view forms support additional data types. This includes `blobs` (DB2, Oracle, and Informix) and `images` (Microsoft SQL Server and Sybase), which map to an attachment field, and several date and time data types, which map to the Date, Time, or Date/Time field types as shown in this section. (For view forms created in previous releases of AR System, Date/Time fields that were mapped to an integer column are still supported.)

In some cases, you can map a different field type to a column type if appropriate for the data. See "Mapping an alternative AR System field type" on page 194.

NOTE

When the data types `nchar`, `nvarchar`, or `ntext` are supported, they are used on Unicode servers, and the `char`, `varchar`, and `text` data types are used on non-Unicode servers.

For information about the database column types used for AR System fields in regular forms, see the *Database Reference*, "Database column types for AR System fields," page 20.

DB2 data type mappings

The following data types are supported for view forms based on a DB2 database table:

DB2 data types	AR System field type
`varchar, varchar2, char, nchar`	Character (limited length)
`clob`	Character (unlimited length)
`integer, smallint, decimal`	Integer
`real, double`	Real
`decimal`	Decimal
`date`	Date
`timestamp`	Date/Time
`time`	Time
`blob`	Attachment field in attachment pool

Microsoft SQL Server data type mappings

The following data types are supported for view forms based on a Microsoft SQL Server database table:

Microsoft SQL Server data types	AR System field types
`nchar,char, varchar`	Character (limited length)
`ntext,text`	Character (unlimited length)
`int, tinyint, smallint`	Integer
`real, float`	Real
`decimal`	Decimal

Microsoft SQL Server data types	AR System field types
`datetime, smalldatetime`	Date/Time (default), Date, Time
`image`	Attachment field in attachment pool

Oracle data type mappings

The following data types are supported for view forms based on an Oracle database table:

Oracle data types	AR System field types
`varchar, varchar2, char, nchar`	Character (limited length)
`clob`	Character (unlimited length)
`number`	Integer
`float`	Real
`number`	Decimal
`date`	Date/Time (default), Date, Time
`blob`	Attachment field in attachment pool

Informix data type mappings

The following data types are supported for view forms based on an Informix database table:

Informix data types	AR System field types
`nvarchar,nchar,char, varchar`	Character (limited length)
`text`	Character (unlimited length)
`int8, smallint, integer`	Integer
`float, smallfloat`	Real
`decimal`	Decimal
`date`	Date
`datetime`	Date/Time (default) or Time
`blob`	Attachment field in attachment pool

Sybase data type mappings

The following data types are supported for view forms based on a Sybase database table:

Sybase data types	AR System field types
`varchar, char`	Character (limited length)
`text`	Character (unlimited length)
`int, tinyint, smallint`	Integer
`float, smallfloat`	Real
`decimal`	Decimal

Sybase data types	AR System field types
`date`	Date
`datetime, smalldatetime`	Date/Time
`time`	Time
`image`	attachment field in attachment pool

Creating and modifying view forms

Use BMC Remedy Developer Studio to create and modify view forms.

The following procedure explains how to create a view form to connect to a database table.

▶ To create a view form

1. If the database is remote, set it up as described in "Setting up a remote database for view forms" on page 194.
2. In BMC Remedy Developer Studio, choose File > New > View Form.
3. In the New View Form wizard, select the server on which you want to create the view form, and click Next.
4. Enter the name of an existing database table to be associated with the view form (see Figure 12-1 on page 193).

   The formats for table names are as follows. Where two formats are given, the first is for a table in the local database and the second for a table in a remote database:

   - **DB2**—`TABLENAME`

     Use all capital letters when entering the table name because DB2 defaults to all capital letters for the data in its system tables.

   - **Informix**—`tablename` or `databasename:table`

     Use all lowercase letters when entering the owner and table name, because Informix defaults to all lowercase letters for data in its system tables.

   - **Oracle**—`TABLENAME` or `OWNER.TABLENAME`

     Oracle defaults to all capital letters for data in its system tables. If the table name uses lower case, make sure that the capitalization for the name is entered correctly.

   - **Microsoft SQL Server**—`TABLENAME` or `LINKNAME.DATABASENAME.OWNER.TABLENAME`

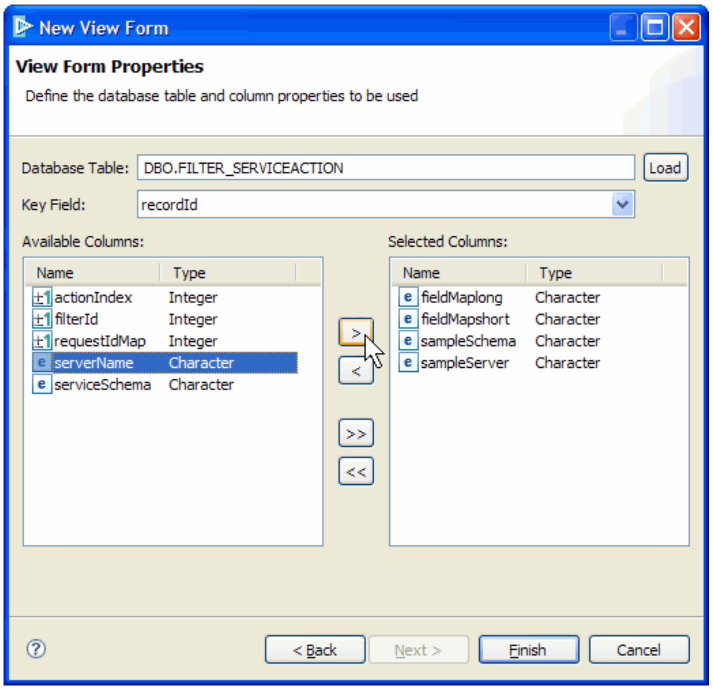
   - **Sybase**—`TABLENAME` or `DATABASENAME.OWNER.TABLENAME`

     Specify the owner as `dbo` if the current user is the owner of the table.

5 Click Load.

The Available Columns list box is populated with the database column names and default AR System field type mappings for the supported data types.

Figure 12-1: New View Form wizard, View Form Properties page



View Form Properties

Define the database table and column properties to be used

Database Table: DBO.FILTER_SERVICEACTION

Key Field: recordId

Available Columns:

Name	Type
actionIndex	Integer
filterId	Integer
requestIdMap	Integer
serverName	Character
serviceSchema	Character

Selected Columns:

Name	Type
fieldMaplong	Character
fieldMapshort	Character
sampleSchema	Character
sampleServer	Character

6 In the Key Field list box, choose a column to designate as the key field.

You must choose either a character column with a name that contains 6 through 15 characters, or an integer column.

WARNING

The selected column must be unique and non-null.

7 In the Available Columns list, select the columns to appear on the AR System form, and use the arrow buttons to move them to the Selected Columns list.

This method maps the default AR System field type to the database data type. To use a different AR System field type for a database column, do not select the column from the list as described in this step. Instead, follow the steps in "Mapping an alternative AR System field type" on page 194.

8 Click Finish.

9 Use Developer Studio to make any additional changes to the new form, and then click File > Save.

Mapping an alternative AR System field type

If necessary, you can map an AR System field type that is different than the default type to a column in the external database.

▶ To map an alternate field type to a database column in a view form

1. Create the view form as described in "To create a view form" on page 192, but in step 7, do not select the column to which you want to map an alternate field type.
2. After saving the form, add a field of the appropriate type to the form.
3. In the field properties tab, expand the View Information properties.
4. Click the Column property and type the database column name.

WARNING

Do not create a form with multiple fields referring to a single column. Such a form will produce adverse results and may generate SQL errors.

Modifying view forms

▶ To add a new field to a view form using the default field type

1. Choose Form > Add Fields From *externalDBName* from the menu.
2. Select the field to add from the Add Field dialog box, and click OK.

▶ To delete a field from a view form

1. Click the field and choose Edit > Delete.

   Deleted fields return to the Available Columns list box. This action does not remove the column from the database table.

Setting up a remote database for view forms

If the database table is in a remote database, you must perform the required database configuration steps described in this section and use the correct table name syntax to access the remote table. For databases that require you to create a database link or proxy database, refer to your database documentation and work with the database administrator to configure access to the remote database table.

- **DB2**—Remote view forms are not supported for DB2.
- **Informix**—When creating the view form, use the following format to access the database table:

  databasename@servername:tablename

- **Oracle**—Set up a link between the AR System database and the Oracle database. Create a view in the database user's schema which accesses this link (the user is ARADMIN by default).

For example:

```
CREATE VIEW view_name AS (SELECT * FROM
ownername.tablename@link)
```

Create the View Form on the view created above.

To enable the support of multiple remote Oracle databases with different character sets, you must add the `Oracle-Dblink-Character-Set` parameter to your `ar.cfg` (`ar.conf`) file. For more information, see the *Configuration Guide*.

- **Microsoft SQL Server**—Complete the following tasks:
  - Create a link to the remote database and either give the user `ARAdmin` an account on the remote database or use the proper login credentials.
  - Turn on the Distributed Transaction Coordinator for both the local and the remote databases.
  - Specify the following server configuration setting in the `ar.conf` (`ar.cfg`) file:

    ```
    SQL-Server-Set-ANSI-Defaults: T
    ```

    This setting enables the DB-Library connection that AR System uses to use ANSI-NULLs and ANSI warnings. There should be no impact on the performance of the database. For more information about the `ar.conf` (`ar.cfg`) file, see the *Configuration Guide*.

  The format for the table name is:

  ```
  LINKNAME.DATABASENAME.OWNER.TABLENAME
  ```

- **Sybase**—Create a proxy database. This is a Sybase database type that copies all the metadata about a remote database to the local database but still allows queries to be redirected to the remote database. For information about how to create a proxy database, see the Sybase documentation.

  The format to access the database table is:

  ```
  DATABASENAME.OWNER.TABLENAME
  ```

Chapter 13 SQL database access

Using SQL, third-party applications can read data from the AR System database. Similarly, both AR System client and server processes can read and write to external databases using SQL.

The following topics are provided:

- Accessing AR System data externally (page 198)
- Pushing data from AR System with SQL (page 198)
- Pulling data into AR System with SQL (page 198)
- Issues and considerations (page 199)

Accessing AR System data externally

Any process that has permission to query the database engine can read AR System data. A third-party application writing to the AR System database is not supported because there is no way to ensure data integrity. In addition, external applications reading AR System data directly from the database are not subject to AR System permissions, nor do they trigger any AR System workflow. If this is not acceptable, data should be read through the AR System API.

For detailed information about the AR System database, see the *Database Reference*.

Pushing data from AR System with SQL

All three AR System workflow components–active links, filters, and escalations–can send data to external tables and even external databases using the Direct SQL action. The SQL command must be created by the administrator and entered into the SQL Command field on the If Action or Else Action tab. The AR System server performs no pre- or post-processing on the SQL command or the results. The administrator must make sure that the command is correct. When the action is triggered, the AR System server passes the SQL command directly to the SQL database server on which it is running. For more information about the Direct SQL action, see the *Workflow Objects Guide*.

Pulling data into AR System with SQL

To pull information from external tables, you can use the Set Fields action with the Read Value for Field From field set to SQL. This allows you to send an SQL `SELECT` command to the database and assign the return values to AR System fields.

Observe the following general rules for using SQL commands:

- You need not use every value that is returned from the SQL command, but you must use at least one.
- You can use the same value in more than one field.
- You can issue only one SQL command per action. You cannot enter two commands separated by a semicolon and have both commands run. To run a set of commands, create separate actions, or create a stored procedure and run that. (Stored procedures do not return values.)
- Turn on AR System server SQL logging to resolve problems with the SQL syntax if it returns unexpected values or results. A good strategy is to start an SQL interpreter (for example, isql for Sybase, SQL*Plus for Oracle, Command Center for DB2, or Query Analyzer for Microsoft SQL Server) and to enter the same SQL command directly into the database to verify its validity.

- Because there is no error checking on the SQL statement, run the SQL statement directly against the database (as a test) before you enter it into the SQL Command field. You can then copy and paste the tested SQL command directly into the SQL Command field.
- If the SQL operation fails, an AR System error message and the underlying database error message appear.

For more information about Set Fields action with SQL, see the *Workflow Objects Guide*.

Issues and considerations

Keep the following issues in mind when working directly with an SQL database:

- The AR System server typically has full administrator access to the database for reading and writing any data. AR System users have permissions to read and write specific data using an AR System client, and these permissions are managed by the AR System server. If users access the database directly through a database client, they are bypassing the AR System security model.
- AR System stores some data in the database in formats that can cause third-party applications to become confused. For example, AR System date / time fields store values as timeticks, which are the number of seconds from 1 January 1970 at midnight until the current time. These numbers are stored as integer numbers, and typically need to be converted by the third-party application.
- All SQL commands are sent to the database server that holds the AR System database. To access databases that are external to this DB server, you must have the appropriate conduit installed and issue the SQL commands needed to use the conduit for your `SELECT` statement.

Chapter

14 ODBC database access

The Open Database Connectivity (ODBC) standard is a connectivity solution that enables ODBC clients to communicate with AR System. The AR System ODBC driver provides read-only access to data defined in AR System forms. This section discusses the use of the AR System Open Database Connectivity (ODBC) driver to provide additional functionality with other programs.

The following topics are provided:

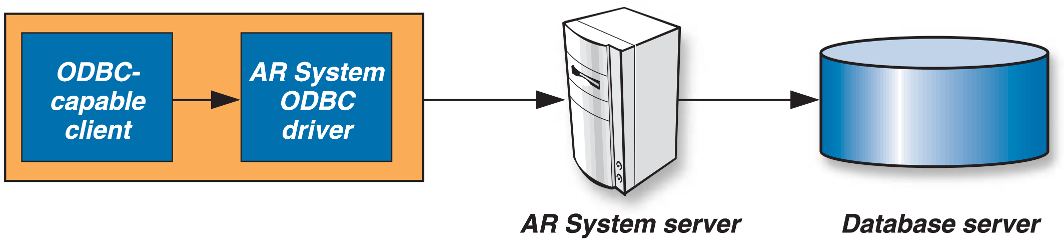
- Overview (page 202)
- Creating multiple data sources (page 202)
- Compatibility with ODBC clients (page 205)
- Using Crystal Reports with AR System (page 205)
- Using Microsoft Access with AR System (page 211)
- Using Microsoft Excel with AR System (page 212)
- Issues and considerations (page 212)

Overview

Open database connectivity (ODBC) is an SQL-based communication standard developed by Microsoft. The ODBC standard represents a connectivity solution that enables ODBC clients to communicate with AR System. The AR System ODBC driver provides read-only access to data defined in AR System forms.

The interface provided by the ODBC driver (`arodbc70.dll`)is similar to that provided by the AR System API. Like the API, the driver does not provide access to the underlying relational database. Instead, as shown in the following figure, the driver communicates with the AR System server, which in turn communicates with the database server. When using the ODBC driver, the AR System access control model (user and group permissions) is maintained, and server-side workflow is still triggered.

Figure 14-1: ODBC integration



Many ODBC clients are available. The AR System ODBC driver provides extended functionality with BusinessObjects Crystal Reports. In addition, the driver provides basic functionality with Microsoft Access, Microsoft Excel, and other ODBC clients. See the compatibility matrix on the BMC Remedy support website for additional information about supported ODBC clients.

Creating multiple data sources

By default, when you install BMC Remedy User, the mid tier, or the ARWebReportViewer, the AR System ODBC driver (`arodbc70.dll`) is installed. The AR System ODBC data source is configured with your AR System user name and password, and it accesses AR System with your permissions. You can designate multiple data sources for one third-party tool; conversely, you can use a single data source for several third-party tools.

For example, to run Crystal Reports with any client (a browser or BMC Remedy User), you must have the AR System ODBC driver on your machine. You could create a data source called Report User to access AR System through Crystal Reports. When you create this data source, you might specify Joe User as the AR System user and supply Joe's password. When you use the Report User data source to access AR System through Crystal Reports, the AR System permissions are granted to Joe User. This enables you to set up data sources with multiple levels of permissions.

▶ To create additional ODBC data sources

1 Open the ODBC Data Source Administrator.

This utility is in different locations based on the version of Windows you use.

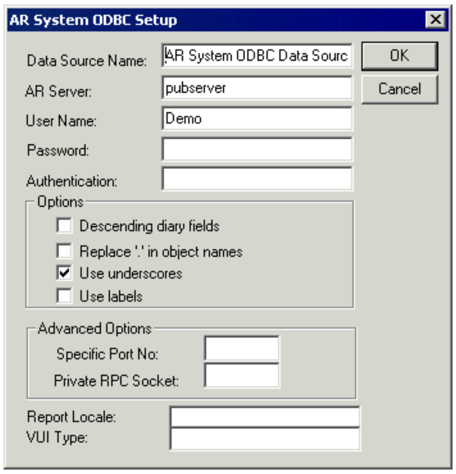
2 On the System DSN tab, select AR System ODBC Data Source, and click Add.

The Create New Data Source dialog box appears.

3 Select AR System ODBC Driver, and click Finish.

The AR System ODBC Setup dialog box appears.

Figure 14-2: AR System ODBC Setup dialog box


AR System ODBC Setup

Data Source Name: AR System ODBC Data Sourc

AR Server: pubserver

User Name: Demo

Password:

Authentication:

Options

- [ ] Descending diary fields
- [ ] Replace '.' in object names
- [x] Use underscores
- [ ] Use labels

Advanced Options

Specific Port No:

Private RPC Socket:

Report Locale:

VUI Type:

4 In the Data Source Name field, enter a unique name for the data source.

5 In the AR Server field, enter the name of the AR System server to access with this data source.

6 Enter a user name whose permissions will be used to access the report data.

7 Enter the user's password.

8 Select the Descending Diary Fields check box to designate reverse calendar order.

9 (Mid tier only) If the field or form names in your reports contain special characters, such as a dot (.), hyphen (-), plus sign (+), and semicolon (;), select the Use Underscores check box to replace the special characters with underscores.

For reports displayed in BMC Remedy User, you do not need to select the Use Underscores check box. BMC Remedy User supplies the correct value.

NOTE

If you use Microsoft Access, spaces and hyphens are not allowed in object names.

10 To use field labels based on the locale you specify in the Report Locale field, select the Use Labels check box. See "Using field labels or database names in Crystal Reports" on page 208.

NOTE

If the Verify On First Refresh option in Crystal Reports is *selected*, you must match the state of the Use Labels option when you create the report and at run time. For example, if you select the Use Labels option when you create the report, you must select it when you run the report. If you clear the Use Labels option when you create the report, you must clear it when you run the report.

To avoid problems caused by mismatched Use Labels options, it is recommended that you *clear* the Verify On First Refresh report option in Crystal Reports.

11 (Mid tier only) In the Report Locale field, enter the locale for the language in which to display the report.

NOTE

If you install two localized views (for example, German and French) and you use the German localized view and the report locale setting is set to the French locale, the data is returned in French, though the static report text is in German.

For reports displayed by BMC Remedy User, you do not need to specify a value in the Report Locale field. BMC Remedy User supplies the correct value.

12 (Mid tier only) In the VUI Type field, enter 3 to specify that a web view should be used to display reports for this data source.

For reports displayed by BMC Remedy User, you do not need to specify a value in the VUI Type field. BMC Remedy User supplies the correct value.

13 Click OK.

NOTE

To modify an existing data source, select it in the ODBC Data Source Administrator dialog box, and click Configure. The dialog box in Figure 14-2 is displayed.

Compatibility with ODBC clients

Many ODBC clients are available. The AR System ODBC driver provides:

- Multi thread-safe operation
- Compatibility with ODBC version 3.5
- Support for Unicode
- Extended functionality with Crystal Reports 10.0 and XI, which enables you to create custom reports with wide-ranging capabilities and provides additional flexibility in report design.
- Basic functionality with Microsoft Access.
- Basic functionality with Microsoft Excel.

See the compatibility matrix on the Remedy website for additional information about supported ODBC clients.

Using Crystal Reports with AR System

After you set up predefined reports using Crystal Reports, users can view them by using BMC Remedy User and the Crystal Reports Display Engine. The Crystal Reports Display Engine is automatically installed with the BMC Remedy User installation. For more information, see the *Installation Guide*. For information about viewing reports created with Crystal Reports, see BMC Remedy User help.

NOTE

Before you start creating reports based on AR System forms, make sure that you follow the SQL standard for naming objects such as forms. For example, start the form name with an alphabetic or underscore character. You should especially avoid using a number (such as 2) for the name of a form. Otherwise you might see an error message, such as `ODBC error: Unexpected extra token: `*`formName`*.

The following procedure describes how to get started designing reports; however, see your Crystal Reports documentation for complete instructions about using the design wizard to create reports.

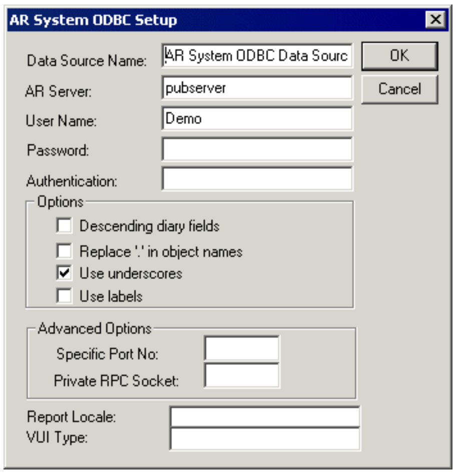
▶ To create a report by logging in to AR System from Crystal Reports

1. Open Crystal Reports and create a report.
2. In the Crystal Reports Gallery, select a wizard such as Standard.
3. In the Available Data Sources, select ODBC (RDO).
4. When the ODBC (RDO) dialog box appears, select the Data Source to log in to.

   For example, select AR System ODBC Data Source as the default data source.

   The AR System ODBC Setup dialog box appears.

Figure 14-3: AR System ODBC Setup dialog box


AR System ODBC Setup

Data Source Name: AR System ODBC Data Sourc
AR Server: pubserver
User Name: Demo
Password:
Authentication:

Options
- [ ] Descending diary fields
- [ ] Replace '.' in object names
- [x] Use underscores
- [ ] Use labels

Advanced Options
Specific Port No:
Private RPC Socket:

Report Locale:
VUI Type:

5 Enter the user's password.

6 To designate reverse calendar order, select the Descending Diary Fields check box.

7 Select the Use Underscores check box.

8 Specify whether to use field labels or database names to represent AR System fields.

- Select the Use Labels check box to use field labels.
- Clear the Use Labels check box to use database names.

See "Using field labels or database names in Crystal Reports" on page 208.

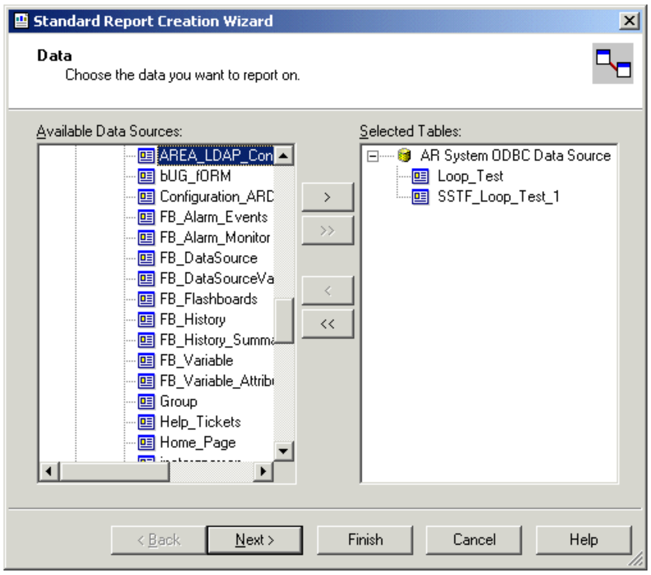
NOTE

Field labels are based on the locale specified in the Report Locale field.

9 Click OK to log in.

The AR System forms appear in the Standard Report Creation Wizard as data sources.

Figure 14-4: Selecting data sources in Standard Report Creation Wizard


Standard Report Creation Wizard

Data
Choose the data you want to report on.

Available Data Sources:
AREA_LDAP_Con
bUG_fORM
Configuration_ARC
FB_Alarm_Events
FB_Alarm_Monitor
FB_DataSource
FB_DataSourceVa
FB_Flashboards
FB_History
FB_History_Summ
FB_Variable
FB_Variable_Attrib
Group
Help_Tickets
Home_Page

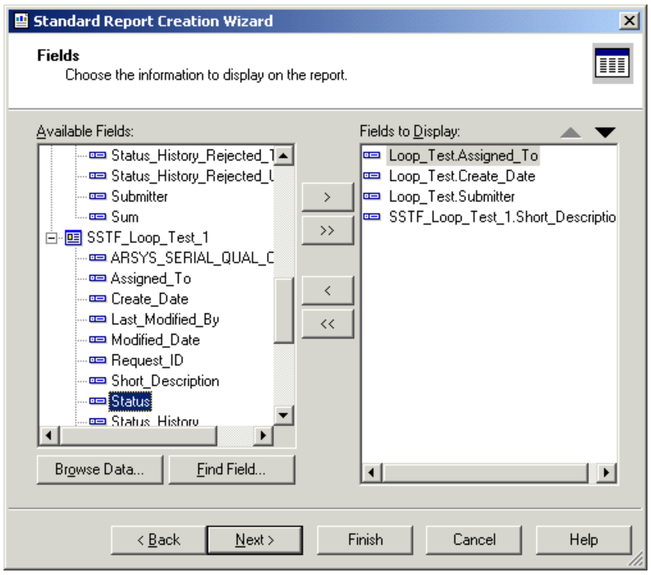
Selected Tables:
AR System ODBC Data Source
Loop_Test
SSTF_Loop_Test_1

10 Select the form to include in your report, and click Next.

11 Click Next.

The wizard lists all fields in the underlying form.

Figure 14-5: Selecting fields in wizard


Standard Report Creation Wizard

Fields
Choose the information to display on the report.

Available Fields:
Status_History_Rejected_1
Status_History_Rejected_L
Submitter
Sum
SSTF_Loop_Test_1
ARSYS_SERIAL_QUAL_C
Assigned_To
Create_Date
Last_Modified_By
Modified_Date
Request_ID
Short_Description
Status
Status_History

Fields to Display:
Loop_Test.Assigned_To
Loop_Test.Create_Date
Loop_Test.Submitter
SSTF_Loop_Test_1.Short_Descriptio

NOTE

The content of the list of fields depends on whether you selected Use Labels in the AR System ODBC Setup or AR System ODBC Login dialog box. See "Using field labels or database names in Crystal Reports" on page 208.

When you select report fields, some fields might not be listed that are in your form. This occurs when the field's database name is different from its display label. For example, a field called Last Name in your form is not shown in the Database Fields list box in Crystal Reports (the Database Fields list box appears in the following figure). Instead, the field name Surname might appear. Each field in a form is identified by a unique database name, not by the display label that appears in the form.

To identify a field's database name, open the form in BMC Remedy Developer Studio, and open the Field Properties dialog box for the field. The Name field of the Database tab in the Field Properties dialog box shows the field's database name.

12 Select the fields to display in the report, and click Next.

The wizard now lets you specify how to group the information to display on your report. This step is optional.

13 Group the information, and click Next.

The wizard now lets you specify a subsection of the information to display on your report. This step is optional.

14 Select a subsection of information, and click Next.

The wizard now lets you specify a report template. This is optional. To preview your report, click an available template.

15 Select a template, and click Finish.

For more information about designing reports, see your Crystal Reports documentation.

Using field labels or database names in Crystal Reports

When you create a report in Crystal Reports, you select the AR System fields on which to report from a list displayed by Crystal Report Designer. The AR System fields in the list are represented by field labels (generated at the AR System view level) or database names (generated at the AR System database level). You specify how to represent AR System fields in Crystal Report Designer and your reports when you:

- Set up the AR System ODBC as a data source for your report. See "To create additional ODBC data sources" on page 203.
- Create a report in Crystal Reports. See "To create a report by logging in to AR System from Crystal Reports" on page 205.

If you specify using field labels, the list of fields in Crystal Report Designer displays field labels, if any exist. If a field does not have a label, the list displays the database name for that field.

If you *do not* specify using field labels, the list of fields in Crystal Report Designer displays database names for the fields.

IMPORTANT

It is recommended that you use the same setting for setting up your AR System ODBC and for creating your Crystal Report.

WARNING

If you report on two AR System fields that have the same label or two fields where one field's database name matches another field's label, your report might contain incorrect data.

TIP

To identify a field's database name, open the form in BMC Remedy Developer Studio, and then open the Field Properties dialog box for the field. The Name field of the Database tab in the FieldProperties dialog box shows the field's database name.

Crystal Report report options considerations

When you create a report in Crystal Reports, you can select the Verify on Refresh option. When this option is selected, Crystal Reports verifies fields defined in a report, which can be either AR System field labels or database names, against the data source as it is configured at run time. If you use this type of verification, Crystal Reports reports only on field names for which there are matches between the report definition and the data source as it is configured at run time.

If you change your AR ODBC configuration (that is, toggle the Use Labels check box) between the time you design the report and the time you run it *and* you verify report fields against the AR ODBC data source, your report might not contain the data you expect, and you might receive a columns-not-found error.

Selecting report fields in Crystal Reports

When using an ODBC client to view AR System data, some fields that are in your form might not be listed. This occurs when the field's database name is different from its display label.

Suppose, for example, a field in your form called Last Name is not shown in the Database Fields list box in Crystal Reports. Instead, the field name Surname might appear. Each field in a form is identified by a unique database name, not by the display label that appears in the form.

TIP

To identify a field's database name, open the form in BMC Remedy Developer Studio. Select the field, then expand the Database category on the Properties tab. The Name property displays the database name.

▶ To select the field

1. In the Create Report Expert dialog box, click the Fields tab.
2. From the Database Fields list, select the fields to include in your report, and click Add.

   Alternatively, you can click Add All to include all the fields. To remove a field or all fields, click Remove or Remove All, respectively.
3. Click Preview Report to view your report.

   For information about designing reports, see your Crystal Reports documentation.

Using Crystal Reports with join forms

Crystal Reports allows users to generate reports from multiple tables by joining the tables together in an SQL statement external to AR System. AR System ODBC Driver does not support this capability. You can, however, achieve the same goal by creating an AR System join form. After creating the join form, generate a report from it.

If you add two fields that have the same database name (such as Submitter) to a join form, one field's database name appears as a field ID in Crystal Reports.

Limitations when using Crystal Reports

Be aware of the following limitations when using Crystal Reports:

- **Converting Date/Time Strings to Date Strings**—In Crystal Reports, you can specify how Date/Time strings are handled in your report. If you select the *Convert to Date* option in the Reporting tab of the File Options dialog box, Date/Time strings from AR System are converted to Date strings in Crystal Reports.

  However, if you set this option to convert Date/Time strings to Date strings, you cannot use the select condition *is equal* (in the Select tab of the Create Report Expert dialog box in Crystal Reports). The AR System Date/Time field works only with the Convert to String or Keep Date-Time Type options.
- **List Sorting**—Selection fields from AR System are treated as character types. List sorting in Crystal Reports is based on display values (New, Assigned, Closed), rather than numeric values (0, 1, 2) associated with an enumerated field. This occurs because selection fields with `AR_DATA_TYPE_ENUM` data types are mapped to `SQL_CHAR` data types when the AR System ODBC driver is used. ODBC does not have an equivalent data type.
- **Browsing Data**—The Browse Data button in the Fields tab of the Create Report Expert dialog box in Crystal Reports does not display the Request ID (or other data) for all the requests. (Do not select the Select Expert option because it attempts to perform an unqualified search for *all* values in a field.)
- **Date**—Crystal Reports follows the calendar type from your operating system, typically the Gregorian calendar starting from October 15, 1582. If the date field contains a BC date, Crystal Reports does not support it.

Using Microsoft Access with AR System

This section includes tips for using Microsoft Access with AR System.

- Avoid using special characters (such as brackets, decimal points, hyphens, and spaces) when naming tables and columns.

  When you set up an ODBC driver for use with Microsoft Access, select the Use Underscores check box. This check box is shown in Figure 14-2 on page 203.

- Table names that are nearly identical, such as `My.Table` and `My Table` (names that include decimal points, hyphens, and spaces), are not differentiated by the driver.

  Searching for data in these tables might produce unexpected results. Rename table and field names that are nearly identical.

- Maximum size of an entry or data set in Microsoft Access is 2K.

  If you encounter the errors `Record too large` when using the Import Table option or `This form or report is based on a query that exceeds the limit for data in a single record` when using the Link Table option, you must exclude unnecessary fields from the search or report. See your Microsoft Access documentation for additional information about excluding fields.

- Your Microsoft Access authorized signature and your AR System user name and password might conflict.

  If you notice that the tables or fields disappear (although you have access permissions) when you work on reports, it is caused by a login identification conflict. To resolve this problem:

  - Select the same user name and password that you use to log in to AR System.
  - Turn off the following flag in the Registry and set the value to 0:
    `HKEY_LOCAL_MACHINE\Software\Microsoft\Jet\3.5\Engines\ODBC\TryJetAuth`

- When using Microsoft Access to link tables from an AR System ODBC data source, you enter information into several dialog boxes. Do not select any options from the Select Unique Record Identifier dialog box. Simply click OK to close that dialog box.

Using Microsoft Excel with AR System

When you create an unqualified search for a diary field in Microsoft Excel, the data appears with small control characters that appear as small boxes.

▶ To remove the control characters

1. Highlight the cells and choose Data > Text to Columns.
2. Select the Delimited option, and click Next.
3. Click the Treat Consecutive Delimiters as One button.
4. Select Finish.

   The diary field text data (not the time stamp) is removed with the control characters.

NOTE

Microsoft Excel has a date system that begins January 1, 1900. If your date field contains a BC date, Microsoft Excel does not support it.

Issues and considerations

Consider these issues when working with AR System ODBC:

- The AR System ODBC driver is read-only. ODBC clients that create, modify, or delete entries or tables do not function correctly with the AR System driver.
- The AR System ODBC presents AR System join forms as a single table, enabling you to search AR System join forms easily. However, in third-party ODBC clients, such as Crystal Reports, you cannot run an SQL search that performs a join directly through the SQL statement.

  If you cannot create an AR System join form for the data you need, it is possible to create multiple AR System data sources, connect to one AR System table per data source, and then perform the join in your ODBC client. (BMC Remedy User does not support multiple AR System data sources. Therefore, if you create a report using a third-party ODBC client and join two tables directly in an SQL statement, the report will not run from AR System workflow or from the BMC Remedy User Reporting window.)
- Hidden form permissions are not enforced in the ODBC driver. Forms that are hidden from the BMC Remedy User Object List are accessible for reporting to other tools using the BMC Remedy ODBC driver.

- If you use the AR System ODBC Driver in MS Access to link tables, you might encounter the following error: `Cannot define field more than once`. As a workaround, select the Use Underscores during the DSN configuration. This option makes form and field names adhere to SQL standards by removing spaces and other nonstandard characters.

  To determine which fields are in conflict, you can enable ODBC Tracing and look through the logs, or you can navigate through the Fields list in BMC Remedy Developer Studio to see if there are fields that meet the preceding conditions.

- When the ODBC driver accesses a currency field, it generates four or more column names for the field by adding suffixes to the field name. The suffixes are:

  - `_Date`
  - `_Type`
  - `_Value`
  - *`_functional_currency_code`*

  The driver creates one column for each functional currency code defined for the field.

  If the form contains a field with a name that is the same as one of the generated names, the ODBC driver will report “Cannot define field more than once” and fail to get the data.

  To prevent this problem, do not use field names that conflict with the column names generated by the ODBC driver for currency fields.

Chapter

15 Extending BMC Remedy Developer Studio

This section explains how to add custom functionality to BMC Remedy Developer Studio.

The following topics are provided:

- About extending BMC Remedy Developer Studio (page 216)
- Prerequisites for creating plug-ins (page 219)
- Extension points (page 219)
- BMC Remedy Developer Studio API (page 220)
- Installation directory (page 220)

About extending BMC Remedy Developer Studio

BMC Remedy Developer Studio is composed of Eclipse plug-ins, which are modules of code that perform various functions. Some of these plug-ins have public *extension points*, which are ports through which they expose their functionality to other plug-ins and indicate which class or method to call to use that functionality. To add functionality to BMC Remedy Developer Studio, you can create custom plug-ins with *extensions* that hook into these extension points. Through these connections, custom plug-ins can exchange API calls with BMC Remedy Developer Studio and the AR System server.

IMPORTANT

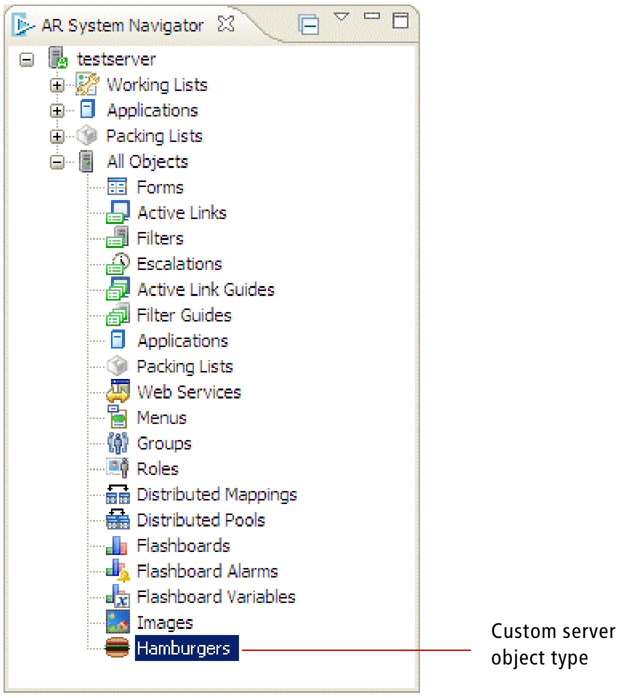
To create plug-ins for BMC Remedy Developer Studio, you must be familiar with Eclipse plug-in development (see `http://www.eclipse.org`) and Java™ (see `http://www.oracle.com/technetwork/java/index.html`). Although BMC Customer Support is available to answer questions about BMC plug-ins and APIs, it cannot provide help with general Eclipse and Java issues that you encounter while developing custom plug-ins.

This feature does not apply to BMC Remedy AR System release 7.5.00 or earlier.

Using the public BMC Remedy Developer Studio plug-in extension points, you can create plug-ins that extend its user interface as follows:

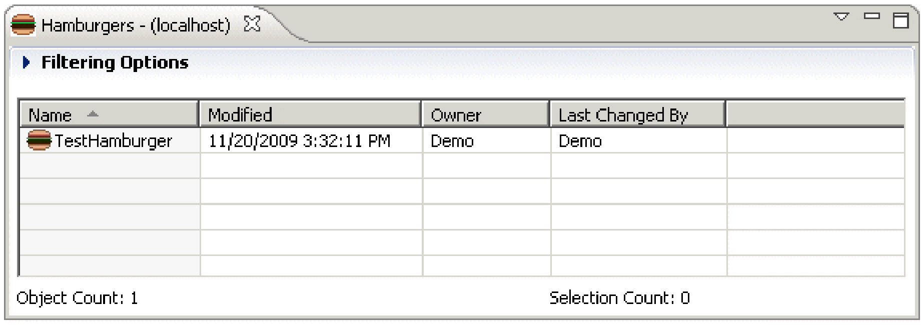
- Add custom server object types to the All Objects list in AR System Navigator and to object lists in the Object List tab. For example, Figure 15-1 shows an All Objects list that contains a custom Hamburgers server object.

Figure 15-1: Customized All Objects list in AR System Navigator



When users double-click the custom object type, an object list for that type opens and displays a list of objects supplied by the custom plug-in (see Figure 15-2).

Figure 15-2: Customized object type in the Object List tab



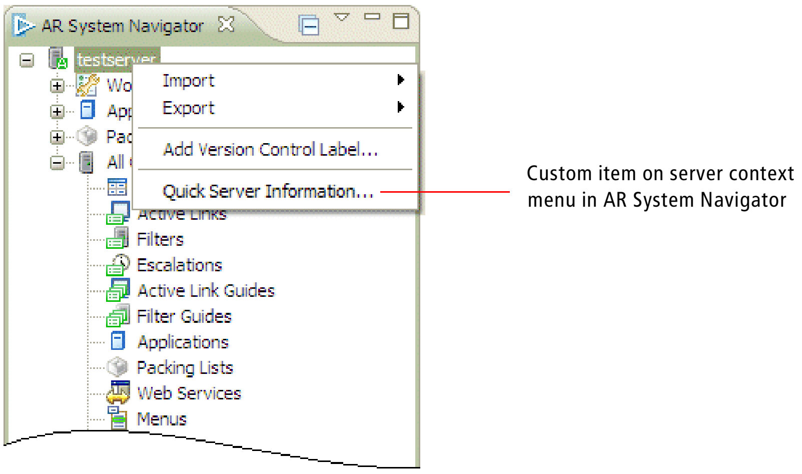
Name	Modified	Owner	Last Changed By
TestHamburger	11/20/2009 3:32:11 PM	Demo	Demo

Object Count: 1   Selection Count: 0

To edit the custom objects, create a custom editor configured by the custom plug-in. In Eclipse, register the editor for the custom object type.

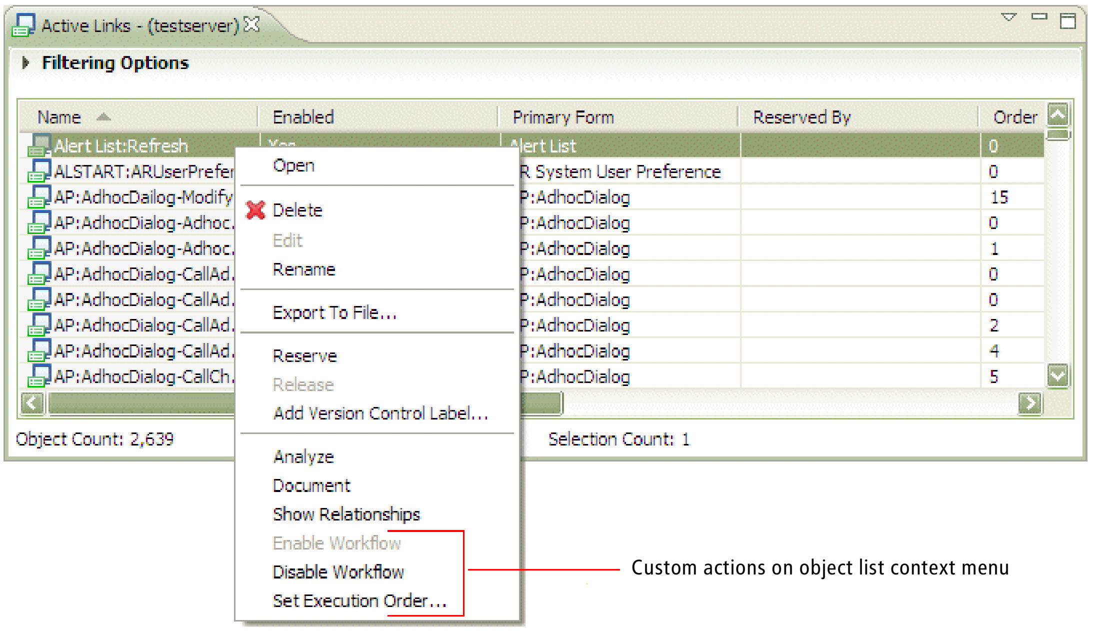
- Add custom items to context menus for servers in the AR System Navigator.

Figure 15-3: Customized server context menu in AR System Navigator



- Add custom items to context menus for object lists. The items can appear on menus for a specific object type or for all object types. For example, Figure 15-4 shows a context menu for active links that contains the following custom actions: Enable Workflow, Disable Workflow, and Set Execution Order.

Figure 15-4: Customized context menu for the Active Link object type



Name	Enabled	Primary Form	Reserved By	Order
Alert List:Refresh	Yes	Alert List		0
ALSTART:ARUserPrefer		R System User Preference		0
AP:AdhocDailog-Modify		P:AdhocDialog		15
AP:AdhocDialog-Adhoc.		P:AdhocDialog		0
AP:AdhocDialog-Adhoc.		P:AdhocDialog		1
AP:AdhocDialog-CallAd.		P:AdhocDialog		0
AP:AdhocDialog-CallAd.		P:AdhocDialog		0
AP:AdhocDialog-CallAd.		P:AdhocDialog		2
AP:AdhocDialog-CallAd.		P:AdhocDialog		4
AP:AdhocDialog-CallCh.		P:AdhocDialog		5

Object Count: 2,639 Selection Count: 1

- Add custom rules to Analyzer.

  You can add rules for the BMC Remedy Developer Studio Analyzer command by creating custom plug-ins that connect to the Analyzer plug-in through its public extension point. (For general information about Analyzer, see the *Workflow Objects Guide*.)

Prerequisites for creating plug-ins

To create plug-ins for BMC Remedy Developer Studio, you need the software and project dependencies listed in this section.

Software requirements

- BMC Remedy Developer Studio release 7.6.02 or later
- Java SE JDK™ 1.5 or later
- Eclipse for RCP / Plug-in Developers (Ganymede 3.4 SR2 or later for Windows)

  To download the Eclipse software, go to `http://www.eclipse.org/downloads/packages/release/ganymede/sr2`.

  Extract the downloaded ZIP file into `C:\Eclipse3.4` (or later version).

Project dependencies

Add the following BMC Remedy Developer Studio plug-ins as dependencies to your custom plug-in project:

- `com.bmc.arsys.studio.api (7.6.02 or higher)`
- `com.bmc.arsys.studio.commonui (7.6.02 or higher)`
- `com.bmc.arsys.studio.model (7.6.02 or higher)`
- `com.bmc.arsys.studio.ui (7.6.02 or higher)`
- `com.bmc.arsys.studio.analyzer.core (7.6.02 or higher)`

To do this, extract the contents of the *`ARSystemServerInstallDir`*`\DeveloperStudio\files\Plugins.zip` file into your top-level Eclipse directory.

For example, if your top-level Eclipse directory is `C:\eclipse`, extract the contents into `C:\eclipse`.

Extension points

The BMC Remedy Developer Studio plug-ins provide these extension points:

Plug-in	Description
`com.bmc.arsys.studio.model.modeltype`	Registers new server object types with the AR System Navigator. Examples of server object types are active links, filters, and forms.
`com.bmc.arsys.studio.model.modelprovider`	Registers providers for new object types. For example, list providers supply a list of objects and object providers supply actual objects for editing. The providers can supply items to both the AR System Navigator and the Object List tab.

Plug-in	Description
`com.bmc.arsys.studio.commonui.typeaction`	Adds actions to context menus in the Object List tab for a single object type. For example, see Figure 15-4 on page 218.
`com.bmc.arsys.studio.commonui.genericaction`	Adds actions to context menus in the Object List tab. The actions can appear on menus for one or more object types.
`com.bmc.arsys.studio.commonui.typeinformation`	Registers new server object types with the user interface to add the types to the AR System Navigator (see Figure 15-1 on page 217) and the Object List tab (see Figure 15-2 on page 217). It also provides the name of the object type to display in the AR System Navigator.
`com.bmc.arsys.studio.analyzer.core.analyzerRules`	Adds rules for the BMC Remedy Developer Studio Analyzer command.

BMC Remedy Developer Studio API

To incorporate additional functionality—such as information about AR System objects and workflow—into custom plug-ins, use the public Java API calls in the `com.bmc.arsys.studio.model` plug-in. These calls are described in the BMC Remedy Developer Studio Java API online documentation, which is in the *`DevStudioInstallDir`*`\files\DevStudioAPIdoc.zip` file.

To access the documentation, unzip the `.jar` file, and open the `index.html` file.

Installation directory

To integrate a custom plug-in with BMC Remedy Developer Studio, put the plug-in's `.jar` file in the *`ARSystemInstallDir`*`\DeveloperStudio\plugins` directory.

Chapter

16 BMC Atrium Integration Engine

The BMC Atrium Integration Engine (AIE) provides the hooks to enable data to pass between AR System and other systems, such as an Enterprise Resource Planning (ERP) system. BMC Remedy provides the structure for developers to build adapters to various databases and products. This chapter provides a brief overview of the AIE integration with AR System.

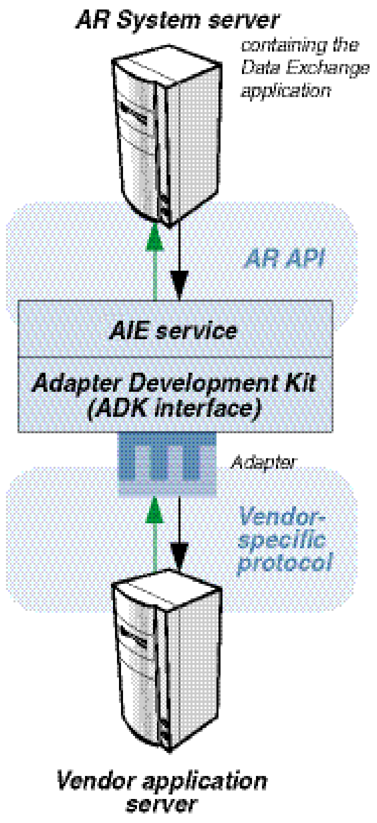
The following topic is provided:

- The AIE integration with AR System (page 222)

The AIE integration with AR System

AIE consists of the Data Exchange application and the AIE service as well as a configuration tool and an event request interface. The Data Exchange application is a group of AR System forms that define how to transfer data between a BMC Remedy application and another application. The AIE service does the actual transfer, using the rules from the Data Exchange application. During the transfer process, the service connects to both the AR System server and to the adapters that communicate with the other application. Figure 16-1 illustrates this process.

Figure 16-1: BMC Atrium Integration Engine


AR System server
containing the Data Exchange application
AR API
AIE service
Adapter Development Kit (ADK interface)
Adapter
Vendor-specific protocol
Vendor application server

The adapters communicate with AIE using the Adapter Development Kit (ADK) interface. The ADK contains the following items:

- A class library defining the interface between the AIE service and the adapter
- An adapter template, which is a development environment that you can use to create an adapter.
- A sample flat file adapter, which is an implementation of an adapter for a flat file database.
- An installation control file to help build an installer for an adapter.

The *BMC Atrium Integration Engine User's Guide* contains information about creating data exchanges and data mappings, and other important AIE concepts.

The *BMC Atrium Integration Engine Adapter Development Kit Developer's Guide* contains detailed instructions for creating adapters.

Chapter

17 BMC Atrium Orchestrator

BMC Atrium Orchestrator (Atrium Orchestrator) enables IT organizations to automate tasks and processes, such as trouble ticketing, fault management, performance monitoring, virtualization management, and so on.

This chapter describes how to configure AR System for integration with Atrium Orchestrator. To use the information in this chapter, you should be familiar with creating forms and workflow in AR System, and you should understand how to use the Set Fields action in a filter or escalation. You must also be familiar with Atrium Orchestrator processes and operations.

The following topics are provided:

- Overview (page 226)
- The AR System Orchestrator Configuration form (page 226)
- AR System workflow for Atrium Orchestrator integration (page 229)

NOTE

To integrate AR System with Atrium Orchestrator, you must use BMC Atrium Orchestrator 7.5 or later. For the latest, most complete compatibility information, see the AR System compatibility matrix on the BMC Support web site at `http://www.bmc.com/support`. For more information about Atrium Orchestrator processes and operations, see your Atrium Orchestrator documentation.

Overview

Application developers can integrate AR System applications with Atrium Orchestrator. This integration requires the AR System Web Services plug-in, which uses the Java plug-in server, to be installed with the AR System server. AR System acts as a consumer of the Atrium Orchestrator web service.

To integrate an AR System application with an Atrium Orchestrator web service, you must complete these two main tasks:

- Create an entry in the AR System Orchestrator Configuration form. Each entry in this form defines the configuration for a specific Atrium Orchestrator web service.
- Create workflow to integrate the application with Atrium Orchestrator, including:
  - A form containing the fields that will hold input and output values for data exchanged with Atrium Orchestrator.
  - A filter or escalation associated with this form. The filter or escalation must include one or more Set Fields actions that use the BMC ATRIUM OCHESTRATOR data source.

NOTE

The tools you use to create AR System workflow and applications and to create and customize Atrium Orchestrator processes have very similar names. This guide describes using BMC Remedy Developer Studio (Developer Studio) to create AR System workflow that integrates with Atrium Orchestrator.
Atrium Orchestrator includes the workflow modeling tool Atrium Orchestrator Development Studio (Development Studio).

For information about using Developer Studio with AR System, see the *Introduction to Application Development with BMC Remedy Developer Studio* guide. For information about using Development Studio with Atrium Orchestrator processes, see the Atrium Orchestrator documentation.

The AR System Orchestrator Configuration form

The AR System Orchestrator Configuration form allows you to define the list of Atrium Orchestrator web services available. Each entry in the form represents the configuration information for an Atrium Orchestrator service and contains all the information required to connect to that service.

The configuration settings that you enter in this form are used when you design the associated filter or escalation. All information required at run time is stored in the filter or escalation.

▶ To define the Atrium Orchestrator web service for AR System

1. Log in to AR System as a user with administrator privileges, using BMC Remedy User or a browser.
2. On the home page, select AR System Administration Console > System > General > Orchestrator Configuration.
3. In the AR System Orchestrator Configuration form, complete the following fields:
   - **Configuration Name**—A unique name that identifies this entry. This is a required field. The name must be unique, and AR System assigns a GUID to the field.

     When you create the associated filter or escalation, BMC Remedy Developer Studio uses this field to display a list of all the entries in this form.
   - **Grid name**—The grid name for the web service. This is a required field.

     When you create the associated filter or escalation, Developer Studio uses the value in this field along with the service URL to obtain the list of Atrium Orchestrator processes. Developer Studio stores the grid name within the workflow action for use when the service is consumed.
   - **Description**—A description of this web service. This is not a required field.
   - **Service URL**—The URL for the web service on the Atrium Orchestrator server, in the format `http://`*`serverName`*`:`*`port`*`/`*`orchContextPath`*`/`*`orchEndPoint`*. This is a required field. Obtain the Atrium Orchestrator context path and end point from the Atrium Orchestrator administrator.

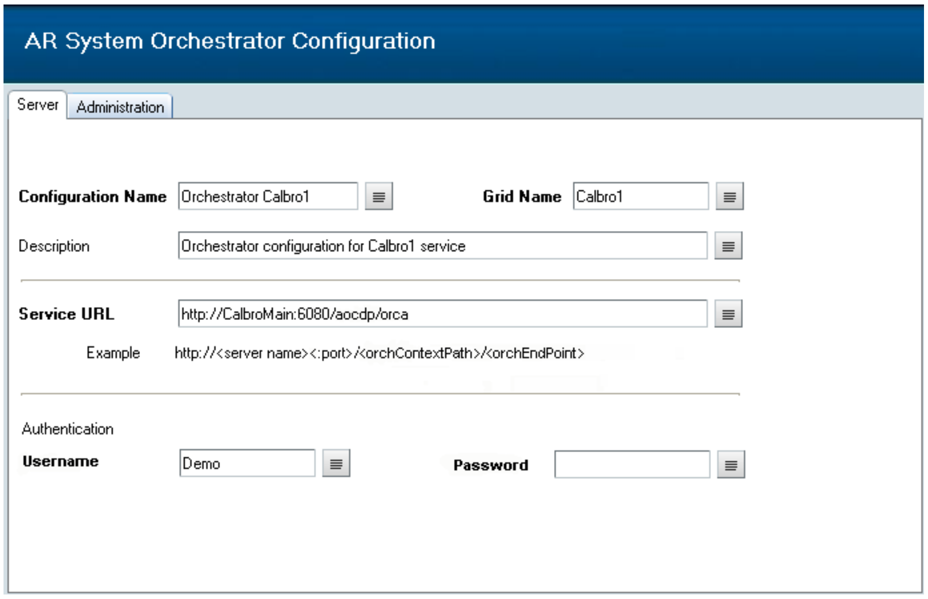
     When you create the associated workflow, Developer Studio uses the value in this field along with the grid name to obtain the list of processes for the service. It also stores the Service URL in the workflow action for use when the service is consumed.
   - **Username**—The user name to use when connecting to the web service. This is a required field.
   - **Password**—The password to use when connecting to the web service. This is a required field.

     Developer Studio uses this user name and password at design time to connect to the web service and obtain data about the service. It also stores this user name and password in the workflow action for use when the service is consumed.

     To override the design-time user name and password with the correct user name and password at run time, you must use the Input Mapping table in the filter or escalation to map these elements to fields in the associated form. See "To define the filter or escalation" on page 230.

   Figure 17-1 shows an example of a completed entry in the AR System Orchestrator Configuration form.

Figure 17-1: Example AR System Orchestrator Configuration form entry



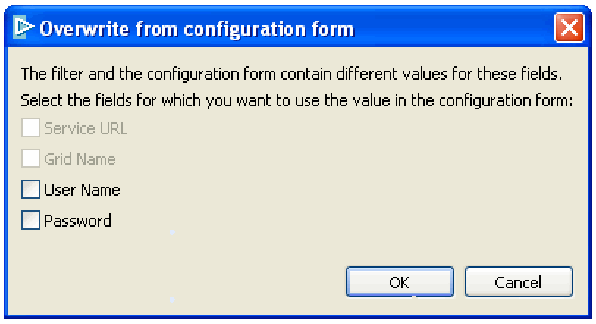
Configuration Name: Orchestrator Calbro1
Grid Name: Calbro1
Description: Orchestrator configuration for Calbro1 service
Service URL: http://CalbroMain:6080/aocdp/orca
Example http://<server name><:port>/<orchContextPath>/<orchEndPoint>
Authentication
Username: Demo
Password:

Modifying entries in the AR System Orchestrator Configuration form

Information stored in the AR System Orchestrator Configuration form is used at design time when you create filters and escalations to perform Atrium Orchestrator operations. When you save the filter or escalation, information from the configuration form is stored in the workflow object.

If you change the information for a configuration entry after associating workflow to the configuration, Developer Studio prompts you to confirm whether you want to override the information stored in the filter with the new information from the configuration form.

Figure 17-2: Overwrite dialog box



The filter and the configuration form contain different values for these fields. Select the fields for which you want to use the value in the configuration form:

- [ ] Service URL
- [ ] Grid Name
- [ ] User Name
- [ ] Password

As shown in Figure 17-2, in the “Overwrite from configuration form” dialog box the active fields are those that have different values in the configuration form entry from those stored in the workflow. To override these values in the workflow object with the values from the configuration form, select the appropriate check boxes. (If the check box is inactive, that value has not changed.) When you save the filter or escalation, the new values are saved with it.

NOTE

If you export the Atrium Orchestrator workflow and import it to another AR System server, you should also export and import the associated entry in the AR System Orchestrator Configuration form. This entry is needed if the filter or escalations that use it are to be edited on the other server. If you do not export the configuration form entry, you can still run the workflow on the other server.

AR System workflow for Atrium Orchestrator integration

To integrate an AR System application with Atrium Orchestrator, use Developer Studio to create an AR System form or forms to hold the input and output data for each process, and a filter or escalation to exchange information with Atrium Orchestrator.

In the AR System filter or escalation, you select the Atrium Orchestrator processes and the operation to perform. The workflow can be designed to carry out either synchronous or asynchronous Atrium Orchestrator operations. With synchronous execution, AR System waits for the operation to complete before returning a result to the workflow action. With asynchronous execution, the operation returns a job ID. In this case, you use the job ID in subsequent workflow actions to determine the status of the operation, or to cancel it.

NOTE

You must use a separate Set Fields action for each Atrium Orchestrator process.

▶ To define the application form

1 Create a regular form, and add the appropriate fields to hold the input and output data for the Set Fields action.

  The input and output fields on the form depend on the operation and process type. You can create one form that will hold the data for all process integrations, or separate forms for each process.

2 Save the form and assign a form name.

▶ To define the filter or escalation

1 Create a new filter or escalation.

2 In the Associated Forms panel, associate the form to the filter or escalation by selecting the form you created in step 1.

3 Select the appropriate Execution Options and enter a Run If qualification that is appropriate for the application.

4 In the If Actions panel, add a Set Fields action with the following settings:

   a As the Data Source, select BMC Atrium Orchestrator

   b In the Configuration Name field, select a configuration from the list.

   Developer Studio obtains the list of available configurations from the entries in the AR System Orchestrator Configuration form. When you select a configuration, Developer Studio retrieves the values for the Service URL and Grid Name, and populates those fields.

NOTE

If you override the values in the Service URL, Grid Name, Username, or Password field, Developer Studio stores the overriding values in the filter or escalation. In this case, whenever the Set Fields action is opened in the future, Developer Studio warns you that the values in the filter do not match the values in the configuration form. At that time you can select which values to replace with those from the form, if necessary.

   c In the Operation field, select an operation from the list.

   The available operation types are:

   - **Synchronous Execution**—AR System waits until the process is complete, and then returns the process result. If the process fails to execute, Atrium Orchestrator returns a SOAP fault and the AR System server reports an error in the filter or escalation.

NOTE

Processes that take longer then 40 seconds to complete cannot be executed in Synchronous Execution mode. If this occurs, AR System reports error 8939: “The AR System Plug-In server is not responding.” For longer processes, use Asynchronous Execution mode instead.

   - **Asynchronous Execution**—AR System returns without waiting for the process to complete. An asynchronous execution operation always returns the Job ID.

- **Cancel Execution**—Cancels the operation identified by the Job ID.

  The time in which you can cancel an operation is limited, based on when the operation started. This is configurable in Atrium Orchestrator. See the Atrium Orchestrator documentation.

  Valid input values for this operation are WITH_COMPENSATION and WITHOUT_COMPENSATION. If you use WITHOUT_COMPENSATION, Atrium Orchestrator returns the job status ABORTED. If you use WITH_COMPENSATION, or if you use an invalid or empty input value, Atrium Orchestrator returns the job status COMPENSATED.

- **Get Job Status**—Returns the current status of the job identified by the Job ID. See "Job status for asynchronous execution operations" on page 232.

d To add an Atrium Orchestrator process to the Process table, click Add.

e In the Add Process dialog box, select a Module from the drop-down list in the Module field.

  A list of Atrium Orchestrator processes appears in the Process list of the Add Process dialog box.

f Scroll through the list of processes and select the appropriate one, then click OK.

  Developer Studio enters the process in the Process table, and populates the Input Mapping and Output Mapping tables with the appropriate Atrium Orchestrator data elements.

g In the Input Mapping table, map each Atrium Orchestrator data element to a field or a static value:

  - To map a field from the associated form, click in the Field / Value column, and then click the ellipsis button. In the Field Selector dialog box, select the field to map to the Atrium Orchestrator data item, and then click OK.
  - To enter a static value, type the value in the Field / Value column.
  - To override the Username and Password stored in the configuration form, map these elements to fields in the associated form, as shown in Figure 17-3, or enter a different static value.

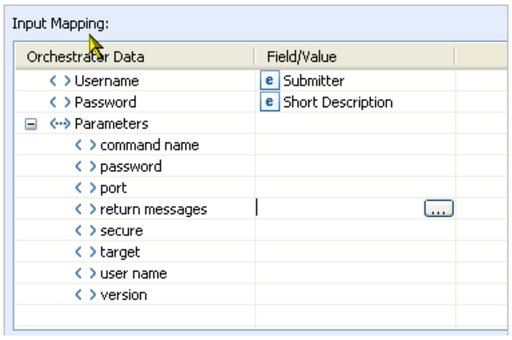
    When you enter a static password value, the plain text password appears in the Field / Value cell until the cell loses focus. From then on, the password value is displayed as a string of asterisks whether or not the cell has focus.

NOTE

The Username and Password from the configuration form are stored in these elements as the `default` attribute, and will be used if the mapped fields are NULL at run time. If you want to prevent this, delete them from the Field / Value column before you map the fields.

Also, Developer Studio automatically sets the attributes `arUsername: true` and `arPassword: true` in these elements. This causes the filter or escalation to use the current user name and password at run time, if no other value is available. You cannot change these attributes.

Figure 17-3: Mapping Username and Password to fields



Input Mapping:

Orchestrator Data	Field/Value
Username	Submitter
Password	Short Description
Parameters	
command name	
password	
port	
return messages	
secure	
target	
user name	
version	

h In the Output Mapping table, map each Atrium Orchestrator data element to a field on the associated form.

NOTE

Output parameters returned to AR System consist of the value only. XML tags generated by Atrium Orchestrator are stripped from the returned value.

Job status for asynchronous execution operations

When the workflow uses asynchronous execution, Atrium Orchestrator returns a job ID. You can use the job ID in subsequent workflow actions to obtain the job status with the Get Job Status operation, and then use the job status to trigger further workflow actions. The possible values for the returned job status are:

- `READY`
- `PENDING`
- `ASSIGNED`
- `IN_PROGRESS`
- `PAUSED`
- `COMPLETED`
- `COMPENSATED`
- `CANCELLED`
- `PENDING_REASSIGNMENT`
- `FAILED`
- `ABORTED`

NOTE

The COMPENSATED status indicates that an error occurred during execution of an asynchronous process. For more information about Atrium Orchestrator job status, see the Atrium Orchestrator documentation.

Chapter 18 Exporting and importing data and definitions

You can export and import objects and data through command-line utilities in place of the graphical user interface.

The following topics are provided:

- Overview (page 234)
- Exporting objects and data to XML format (page 234)
- Using the import/export command-line utility (page 235)
- Using the runmacro command-line utility (page 240)
- Using the BMC Remedy Data Import utility (page 243)
- Using the BMC Remedy User CLI (page 257)

Overview

You can export and import definitions and data in various ways in AR System. *Definitions* are descriptions of the structure in which objects, views, and applications in AR System are organized, identified, and manipulated in the AR System server. Object definitions contain no user data or entries. *Data* is extracted from requests in AR System.

- To export and import definitions, use any of the following options:
    - BMC Remedy Developer Studio menu options (See the *Form and Application Objects Guide*.)
    - The `DefinitionImport.bat` and `DefinitionExport.bat` command-line utilities (See "Using the import/export command-line utility" on page 235)
    - The C and Java APIs (See the *C API Reference*.)
- To export data, use either of the following options:
    - BMC Remedy User (See BMC Remedy User help.)
    - The `runmacro` command-line utility (See "Using the runmacro command-line utility" on page 240.)
- To import data, use either of the following options:
    - BMC Remedy Data Import (See the *Configuration Guide*.)
    - The BMC Remedy Data Import command-line utility (See "Using the BMC Remedy Data Import utility" on page 243.)

This chapter describes the three command line interfaces for importing and exporting data and definitions.

Exporting objects and data to XML format

To prepare to import objects or data, you can export them to an XML file.

AR System objects in XML

Choosing the AR System XML format (`ARXML`) for exported objects produces an XML document that is comparable to the AR System definition file format. It is designed to follow the syntax of the XML specification 1.0.

Specifically, every AR System object type has an associated structure definition in XML, which is specified by the XML Schema Definition (`*.xsd`) file. The `*.xsd` files reside on the AR System server and are used to validate the AR System object definitions as valid XML.

Exported objects in XML format comprise an XML document, which might also be referred to as an instance of a particular XML schema definition for that object. If the XML schema definitions are loaded into an XML editor, someone who is knowledgeable about AR System objects and XML can edit the XML document.

The XML schema definitions are designed to be similar to the definitions in the `*.def` files. For more information about the XML Schema definitions of AR System objects, see the data structure information in the *C API Reference*.

AR System XML definition files are used the same way as the classic `.def` (definition) files. When exporting objects in BMC Remedy Developer Studio, you can choose AR XML Definition Files (`*.xml`) in the Save as type field of the Export File dialog box. The Import File dialog box works in the same way, allowing you to bring in XML definitions to your AR System server.

AR System data in XML

To export in XML, create a report in BMC Remedy User or the web as you normally do. When you run the report or save it to a file, select `ARXML` as the file type. The data is now ready to be manipulated with your XML editor or imported into your XML-compatible applications.

To import XML data, run BMC Remedy Data Import. Open your XML data file by selecting AR XML Files (`*.xml`) in the Files of Type field. The other mapping and import steps are the same as previous versions of AR System Import tool.

Using XML with the AR System API

AR System includes XML schema definitions and API calls that you can use to transform XML and AR System objects. The AR System API calls involving XML are divided into two categories:

- `ARGet` calls, which transform XML objects into AR System structures.
- `ARSet` calls, which transform AR System structures into XML objects.

These calls use the AR System API structures that are described in the `ar.h` file. For more information about the XML API calls, see the *C API Reference*.

Using the import/export command-line utility

This section explains how to use the import/export command-line utility to import and export definitions.

WARNING

If the AR System server has Record Object Relationships enabled, the relationships are recorded as the objects are created during import. If you import a large application or many object definitions, the server might become highly loaded and unresponsive for a period of time.

Guidelines for using the import/export utility

Use the following guidelines to run commands from the import/export command-line utility. Commands and options are listed in "DefinitionImport and DefinitionExport options" on page 236.

- On a computer with BMC Remedy Developer Studio installed locally, set the current directory to the directory that contains the BMC Remedy Developer Studio executable (by default, `C:\Program Files\BMC Software\ARSystem\developerstudio\`). The commands must be run in this directory. File arguments without a directory path are created in the current directory.
- Command-line import and export are provided by the `DefinitionImport` and `DefinitionExport` commands. These commands are implemented as Windows batch (`.bat`) files.
- Every command executed opens a separate AR System session, executes the command, and logs out.
- `DefinitionImport` and `DefinitionExport` parse options in the order they are listed in "DefinitionImport and DefinitionExport options" on page 236. Options are interpreted in a predictable order and might not be executed in the order you enter them.
- You cannot perform an action against two servers with one command. For example, you must issue two commands to export object definitions from one server and import them into another server.
- Enclose arguments that contain blank spaces or symbols in quotation marks.

DefinitionImport and DefinitionExport options

The options in the following table are optional unless the description states they are required. Options that can be repeated are marked in the Repeat column.

Option	Parameter	Description	Repeat
`--version`		Writes the version to standard output and exits without executing other options.	
`-u`	*userName*	Uses the account to connect to the server. Required.	
`-p`	*password*	Uses the password to connect to the server. Required if the account has a password.	
`-x`	*server*	Connects to the server to import or export. Required.	
`-w`	*authentication*	Uses external authentication string or Windows domain to connect to the server.	

Option	Parameter	Description	Repeat
`-portnum`	*portNumber*	Uses the TCP port number to connect to the server. Required if the server does not use the default port and there is no port mapper.	
`-e`	*fileName*	Import from or export to the file. Required.	
`-o`	*logFileName*	Write log and error output to the file. If not specified, the output is written to the standard error output.	
`-inplace`		Import in place. Overwrite each existing object without deleting the object first. DefinitionImport only.	
`-lock`	*lockType lockKey*	Export the objects locked. Valid *lockType* values are ■ **1**—Read only. ■ **2**—Hidden. The *lockKey* is a string used to enforce locking. DefinitionExport only.	
`-a`	*activeLinkName*	Import or export the active link.	+
`-A`		Import or export all active links. Any -a options are ignored.	
`-b`	*DSOPoolname*	Import or export the DSO pool.	+
`-B`		Import or export all DSO pools. Any -b options are ignored.	
`-d`	*DSOMappingname*	Import or export the DSO mapping.	+
`-D`		Import or export all DSO mappings. Any -d options are ignored.	
`-f`	*formName*	Import or export the form.	+
`-F`		Import or export all forms. Any -f options are ignored.	
`-g`	*activeLinkGuideNa me*	Import or export the active link guide.	+
`-G`		Import or export all active link guides. Any -g options are ignored.	
`-h`	*filterGuideName*	Import or export the filter guide.	+
`-H`		Import or export all filter guides. Any -h options are ignored.	
`-k`	*packingListName*	Import or export the packing list.	+
`-K`		Import or export all packing lists. Any -k options are ignored.	

Option	Parameter	Description	Repeat
`-l`	*commandFileName*	Import or export the objects specified by the XML command file. To create an XML import/export command file from a working list or a packing list, use the Save as Import/Export Commands pop-up menu command for packing lists or working lists in BMC Remedy Developer Studio. See the *Introduction to Application Development with BMC Remedy Developer Studio.* You can also use an XML file created from a packing list in an earlier release using the Generate XML command in BMC Remedy Administrator. (This product is no longer available.) Any other object type options are ignored.	
`-m`	*menuName*	Import or export the menu.	+
`-M`		Import or export all menus. Any -m options are ignored.	
`-n`	*applicationName*	Import or export the application.	+
`-N`		Import or export all applications. Any -n options are ignored.	
`-q`	*escalationName*	Import or export the escalation.	+
`-Q`		Import or export all escalations. Any -Q options are ignored.	
`-t`	*filerName*	Import or export the filer.	+
`-T`		Import or export all filters. Any -t options are ignored.	
`-z`	*webServiceName*	Import or export the web service.	+
`-Z`		Import or export all web services. Any -z options are ignored.	

Import/export examples

The following are examples of common tasks you might perform with `DefinitionExport` and `DefinitionImport`.

Exporting objects from an AR System server

When you export objects from the server, you export objects to a target file. You can export all objects for *all* forms by using the following command format:

```
DefinitionExport -u userName [-p password] -x serverName
-e targetFileName -F
```

To export objects from a single form, use the following command format:

```
DefinitionExport -u userName [-p password] -x serverName
-e targetFileName -f formName
```

To parse an XML packing list, and export all objects defined in that packing list, use the following command format:

```
DefinitionExport -u userName [-p password] -x serverName
-e targetFileName -l packingList.xml
```

NOTE

You *cannot* export server objects that include a percent sign (%) in their name.

Importing objects into an AR System server

When you import objects into the server, you import objects from a source file. You can import all objects for all forms by using the following command format:

```
DefinitionImport -u userName [-p password] -x serverName
-e sourceFile -F
```

To import specific objects from a source file, use the following command format:

```
DefinitionImport -u userName [-p password] -x serverName
-e sourceFile -f formName -a activeLinkName
```

To parse an XML packing list, and import all objects defined in that packing list, use the following command format:

```
DefinitionImport -u userName [-p password] -x serverName
-e sourceFile -l packingListName.xml
```

The `-l` option parses the XML packing list and imports all objects defined in the packing list. This option overrides other options in the same command.

Using the runmacro command-line utility

The AR System server includes the `runmacro` utility, which can run a macro or export data as a background process without a GUI. The `runmacro` utility can be run from filter or escalation workflow or as a standalone process (that is, a Windows batch file or a UNIX script). Third-party applications can use the `runmacro` utility to run AR System macros. Because `runmacro` functionality provides no GUI support, it can execute processes that run in the background, but it cannot perform tasks such as displaying a results list.

To run the `runmacro` utility, you must set the library path to the directory where the `runmacro` executable resides. To do so, use these commands:

- Solaris and Linux

  ```
  LD_LIBRARY_PATH=runmacroDir
  ```

- HP-UX

  ```
  SHLIB=runmacroDir
  ```

- AIX

  ```
  LIBPATH=runmacroDir
  ```

The `runmacro` command has the following formats. Items between square brackets are optional. Enclose arguments that contain blank spaces or symbols in double quotation marks.

- You can use the original version of `runmacro` without the output file option (`-o`):

  ```
  runmacro [-h homeDir] [-d macroDir]
  [{-x serverName} ...] { -e | -i } macroName
  [-p parameter=value ...] [-U userName] [-P password]
  [-Q internalQualificationFormat]
  [-q clientToolQualificationFormat]
  [-Z internalFormatQualificationFileName]
  [-z clientToolFormatQualificationFileName]
  [{-w | -W } externalAuthenticationString] [-a portNumber]
  [-O]
  ```

- You can use `runmacro` with the `-o` option to use the `arcopy` syntax, which copies the output to a file:

  ```
  runmacro -o outputFileName [{-x server} ...] -U user]
  [-P password] [{ -f | -s} form] [-t {arx|csv|xml}]
  [-Q internalQualificationFormat]
  [-q clientToolQualificationFormat]
  [-Z internalFormatQualificationFileName]
  [-z clientToolFormatQualificationFileName]
  [{-w | -W } externalAuthenticationString] [-a portNumber]
  ```

When you use the `-o` option to export data with attachments, the attachments folder is created in the same directory as the export file. The attachments folder name uses an integer time stamp (for example, `917732184`), and the folder location is specified in the output file name of the `runmacro` command.

When creating macros, you can record a login with the proper permissions if you perform actions that require those permissions (for example, an administrator deleting records). If your macro does not record a login, you must specify login information using the `-U` option and the `-P` option (if necessary).

This table lists the `runmacro` options, which can appear in any order in the command line:

Option	Description
`-o`	Output file name—Name of the file in which to store the data. The file is initially truncated, and then all the data is written to the file (one data set after another).
`-h`	Home directory—Path to the *`ARSystemHomeDir`* directory. If you do not specify the `-d` option, `runmacro` also looks in this directory for the `arcmds` directory that contains the macro to run. You can create separate home directories for each user whom you want to run a macro. To run a user's macros, copy the user's home directory from the machine where the user runs BMC Remedy User to the Windows server, and specify it with the `-h` option, or use the `-h` option to point to the user's home directory on the machine where the user runs BMC Remedy User.
`-d`	Macro directory—Directory that contains the macro if your macro is not in the *`ARSystemHomeDir`*`\arcmds` directory or if you do not have an *`ARSystemHomeDir`* directory.
`-x`	Server name—Name of a server to connect to. This option might be included more than once to connect to multiple servers. Use the following format: `-x` *`serverName`*
`-e` (or `-i`)	Macro name—Specifies the macro to run.
`-p`	Parameter—Value for a parameter. There might be more than one `-p` option in a command line. If the macro specified (using the `-e` or `-i` options) has a parameter, a value can be supplied by naming that parameter and assigning a value. If the parameter name or value includes a space or other special character, the data must be enclosed in quotation marks to cause proper interpretation of the special characters. Use the following format for each parameter specified: `-p` *`parameter=value`*
`-U`	User name—Required login parameter that identifies the user account. The `-U` option must be in uppercase.
`-P`	Password—Optional login parameter that identifies the user account. Omit the `-P` if the user account has no password. The `-P` option must be in uppercase.
`-Q`	Internal qualification format—Query in AR System internal format.

`-q`	Client tool qualification format—A regular query such as you would use in the BMC Remedy User advanced search bar.
	
	Within the query string, double quotation marks must be preceded by a backslash (`\`), which functions as an escape character. For example:
	`runmacro.exe -o <outputFileName>-x <serverName>`
	`  -U <userName> -P <password> -f <form> -t {arx|csv|xml}`
	`  -q "'Submitter'=\"tester\" AND ('Create Date' >`
	`  \"5/9/2007\" AND 'Create Date' < \"5/16/2007\")"`

Option	Description
`-Z`	Internal format qualification file name—File name containing the query in Remedy internal format.
`-z`	Client tool format qualification file name—File name containing a regular query, for example, like you would use in the advanced search bar in BMC Remedy User.
`-w` (or `-W`)	Authenticator—Name of the external authentication string or Windows NT domain. This is related to the Login window's Authentication field, which is discussed in the *Configuration Guide*.
`-a`	Port number—Port number to which to connect the server.
`-f` (or `-s`)	Form name—Form that is exported. The `-f` (or `-s`) option can be repeated multiple times if there are several forms to export. If multiple servers are selected, each server is searched for the form, and the first one found is all that is exported. To control this, specify only one server environment for the operation. If the `-f` (or `-s`) option is not specified, the system exports *all* available regular data forms. It does not export join or external forms.
`-t`	Type of file to write—File type for the output file: `arx`, `csv`, or `xml`. If not specified, the default of `arx` is used.
`-O`	Forces override—If the user has already logged in as this same user from a different IP address, this option tells the server to use the new IP address of the runmacro client and invalidates the old IP address. Note: This option does not apply to users with administrator permissions.

runmacro example

Assume that you have a Human Resources (HR) application that runs on a Windows machine. When a new employee record is created in the HR application, you want to issue a Service Request to the help desk to set up an office for the employee. Assuming the HR application has the ability to issue a command when the new record is created, you would perform the following procedure.

▶ To set up an office for an employee

1 Copy the `runmacro` utility onto the HR application machine. Assume that it is in the *`ARSystemServerInstallDir`* directory.

2 Record an AR System macro that takes a series of parameters and submits a new Service Request record. Assume that this macro is called `SubNewServReq`.

For information about recording macros, see BMC Remedy User help.

3 Create a script file that the HR application calls when a new employee record is created. The script contains a command such as this:

```
C:\arsystem\runmacro -x server3 -h \arsystem\macros
-e “SubNewServReq” -p "Submitter"="HR" -p "Employee
Name"="$EmpName$" -p "Employee ID"=EmpID -p "Employee
Type"=EmpType -p "Room Number"=RoomNum
```

This command would perform these tasks:

a Take the EmpName, EmpID, EmpType, and RoomNum parameters from the HR application, and use a fixed Submitter ID of HR.

b Substitute them into the parameters in the "SubNewServReq" macro stored in the HR application directory.

c Connect to the AR System server called `server3`.

d Create a Service Request according to the macro definition.

Using the BMC Remedy Data Import utility

Use the BMC Remedy Data Import command-line utility (a Java utility) to automate importing data in a multi- or single-threaded environment. (See "Importing in a multithreaded environment" on page 253.)

You can import with or without a mapping file. See "Importing with a mapping file" on page 255 and "Importing without a mapping file" on page 255.

▶ To use the BMC Remedy Data Import utility on Microsoft Windows

1 Open the `DataImport.bat` file in *`ARSystemInstallDir`*`/dataimporttool`.

2 Edit the batch file to set the following environment variables:

- `APIDROP`—The location where `arapiext`*`vers`*`.jar` and `arapi`*`vers`*`.jar` are installed.
- `JAVA_HOME`—The location of your JDK (for example, `C:\Program Files\Java\jdk1.5.0_07`).
- `Path`

For example:

```
set APIDROP=.\plugins\com.bmc.arsys.studio.api_7.6.04\lib
if not exist "%JAVA_HOME%" set JAVA_HOME=jdkPath
set PATH=%JAVA_HOME%\bin;%PATH%;%APIDROP%
```

3 Add the appropriate options to the command line in the batch file. Make sure the `.jar` file names in the `classpath` reflect the appropriate release of AR System, for example:

```
java -classpath
%APIDROP%\arapi7604.jar;%APIDROP%\arapiext7604.jar;
.com.bmc.arsys.apiext.data.DataImport [options]
```

For a list of available options, see "Options for BMC Remedy Data Import command-line utility" on page 245.

4 At the command line, run the batch file.

▶ To use the BMC Remedy Data Import utility on UNIX

1 Navigate to the *ARSystemHome*/api/lib directory and make sure that the following .jar files, required to run the Data Import Utility, are present in the lib folder:

- arapi*vers*.jar
- arapiext*vers*.jar
- log4j-1.2.14.jar

For example:

- arapi7604.jar
- arapiext7604.jar
- log4j-1.2.14.jar

2 Create a DataImport.sh file in the *ARSystemHome*/api/lib directory.

3 Set the following environment variables in the DataImport.sh file:

- APIDROP—The location where arapiext*vers*.jar and arapi*vers*.jar are installed.
- JAVA_HOME—The location of your JDK (for example, /usr/Java/jdk1.5.0_14).
- Path

For example:

```
set APIDROP=ARSystemHome/api/lib
if not exist "$JAVA_HOME" set JAVA_HOME=jdkPath
set PATH=$JAVA_HOME/bin:$PATH:$APIDROP
```

NOTE

Either execute the following command from the *ARSystemHome*/api/lib directory or set the Library Path and the Path variable using the following command:

```
export LD_LIBRARY_PATH=$ARINSTALL/api/lib:$ARINSTALL/
bin:$LD_LIBRARY_PATH
export PATH=$ARINSTALL/api/lib:$ARINSTALL/bin:$PATH
```

4 Enter the following command to use the Data Import utility on UNIX:

```
java -cp .$APIDROP/arapi7604.jar:.$APIDROP/
log4j-1.2.14.jar:.$APIDROP/arapiext7604.jar
com.bmc.arsys.apiext.data.DataImport [options]
```

NOTE

For a list of available options, see “Options for BMC Remedy Data Import command-line utility” on page 245.

For example:

```
java -cp $APIDROP/arapi7604.jar:$APIDROP/
log4j-1.2.14.jar:$APIDROP/arapiext7604.jar
com.bmc.arsys.apiext.data.DataImport -u userName -p password -x
serverName [-a portNumber] -o dataFilePath [-M
fullyQualifiedMappingFileName] [-l logFilePath]
```

Options for BMC Remedy Data Import command-line utility

Use the following format in the command line. Items between square brackets are optional.

```
com.bmc.arsys.apiext.data.DataImport -u userName -p password
-x serverName [-w externalAuthenticationString] [-r rpcNumber]
[-a portNumber] [-custom custom_optionsFilePath]
[-f destinationFormName] -o dataFilePath
[-filelist listOfFiles] [-z options.xmlFilePath]
[-threads numberOfThreads] [-l logFilePath]
[-e duplicateField] [-b bulk size] [-g activateBulkMode]
[-n suppressFilters] [-t multiMatchOption] [-v] [-i]
[-D duplicateIDOption] [-q option] [-c option] [-h option]
[-charset name] [-M fullyQualifiedMappingFileName]
```

NOTE

`[-m mappingFileName] [-d directoryWithMappingFile]`

Enclose arguments that contain blank spaces or symbols in double quotation marks.

The following table describes available options.

Option	Description
`-u`	User name—Required login parameter that identifies the user account. Note: Every cross-platform CLI command opens a separate BMC Remedy Data Import session, executes the command, and logs out. Therefore, you must log in with every command. If BMC Remedy Data Import does not find the user name, BMC Remedy Data Import prints the usage messages and exits.
`-p`	Password—Password for the user account. If the user account has no password, omit the `-p` option.
`-x`	Server name—The server to log in to. If you do not specify the `-x` option, the server name in the mapping file is used.
`-w`	Authentication string—Name of an external authentication string or Windows NT domain. This is related to the Login window's Authentication field, which is discussed in the *Configuration Guide*, "Setting up an authentication alias," page 76.

Option	Description
`-r`	RPC program number—Private server, for example, if a dedicated import server is available. If you do not specify the `-r` option, the default administrator server's RPC program number (390600) is used.
`-a`	TCP port number—Port number for the server. This value is especially important in a multiple server environment. The option also identifies a TCD specific port, if chosen.
`-custom`	Path to `custom_options.xml`—Specifies the path to the `custom_options.xml` file, which is used to specify date and time formats, the separators to be used, and other information. You can use this option if the source data file is in CSV or ASCII format. See "Using the custom_options.xml file" on page 253. Note: During installation, a sample empty `custom_options.xml` file is installed in the installation folders for BMC Remedy Developer Studio and BMC Remedy Data Import. You can change the name and store it anywhere.

`-f`	**Importing with a mapping file**
	Destination form name—Name of the form to import data into.
	If you do not specify the `-f` option, the form specified in the mapping file is used.
	Importing without a mapping file
	Destination form name or pair.
	A single name indicates that the form name in the source data file matches the form name on the server.
	To specify a pair of names, separate the form names with an equal sign, without any spaces around the equal sign: `"`*`destinationForm`*`"="`*`fileForm`*`"`.
	The destination form is the form on the server into which data is imported. The file form is the form specified in the data file.
	Specifying pairs maps data from one form (specified in the data file) to a different form (identified on the server).
	You can specify multiple pairs by using this option multiple times, for example:
	`-f "`*`Target_form_a`*`"="`*`File_form_b`*`"`
	`-f "`*`Target_form_c`*`"="`*`File_form_d`*`"`
	If the `-f` option is not specified, BMC Remedy Data Import tries to import all data sets in the source data file. For each data set, if a matching destination form is found on the server, the data is imported. If no matching form is found, the data set is ignored.

Option	Description
`-o`	One of the following paths: ■ Path to a directory of data files—For *multithreaded* environments, the `-o` option specifies the path to the directory that contains the data files to import. ■ Path to data file—For *single-threaded* environments, the `-o` option specifies the file that contains the data files to import. If you do not specify the `-o` option, the data file specified in the mapping file is used.
`-z`	Path to the `options.xml` file. Specifies the path to the `options.xml` file, which contains the data import commands and the import parameters for individual files and directories for multithreaded import. The BMC Remedy Data Import command-line utility uses the `-z` option to identify the location of the `options.xml` file. See "Using the options.xml file" on page 250. Note: The `-z` option cannot be used in the `options.xml` file; if used, the BMC Remedy Data Import command-line utility disregards this option.
`-filelist`	(For multithreaded environments only) List of data files to import. The data files can be of any type (for example, `ARX`, `CSV`, `XML`, and `ASC`). All of the data is imported into the form designated with the `-f` or `-M` option. If you do not specify the `-filelist` option, the command imports *all* the data files in the directory specified with the `-o` option, regardless of file type.
`-threads`	(For multithreaded environments only) Number of threads in the pool. Note: Make sure that your hardware configuration can handle the number of threads that you enter. If you do not specify the `-threads` option, the default of 50 threads is used. If the number of files in the data directory or number of files you specify in the `-filelist` option is *less than* the `-threads` value or the default value (50), the number of threads used is equal to the number of files in the data directory or number of files specified in `-filelist` option.
`-l`	Full path name of the log file—Use this option to log details of the import execution.

Option	Description
`-e`	Duplicate field—Field ID of the field to check for duplicate data. For example, for the Short Description field, enter `8`. To specify multiple values for a single schema, separate them with commas and double quotation marks (for example, `"2,4,8"`). The maximum number of IDs you can specify is 6. From this release of BMC Remedy AR System, you can now specify multiple values for multiple schemas. For this, separate the schema names (and their values) by a semi-colon and the values by commas (For example, `SchemaName=1,2,3; SchemaName=4,7,8`). The maximum number of IDs you can specify is 6. Make sure that the source data file includes values for all the fields that are being used for checking duplicate data. When `-e` option is omitted, then Request ID field (field ID 1) is used. Additionally, if the `-e` option is not used for importing records, the BMC Remedy Data Import utility uses bulk mode to import records. (See the `-b` option for more information about the bulk size.)
`-b`	Bulk size—Specifies the number of records to process in bulk simultaneously. For AR System 7.1.00 and later versions, the default size is 100. Note: If the `-e` option is used, records are imported individually. If the value of the `-b` option is set to `0`, bulk mode will not be used.
`-g`	If the `-e` option is used, the bulk transaction mode is switched off by default. In this case, you can still activate the bulk transaction mode using the `-g` option. Note: The bulk transaction mode is purposefully switched off with `-e` option as it gives different results than sequential import when there are duplicate records within the data file itself. To forcefully use bulk mode when the `-e` option is used you can use the `-g` option, introduced in this release of BMC Remedy AR System. You can decide whether you want to use the `-g` option when there are duplicate records in your data files. For example, you must use the `"-g 1"` value in the Java Import command line to use the bulk mode. If any value other than `"-g 1"` is used, the force bulk mode is not activated.
`-n`	Suppress filters—Suppresses the merge filters during merging of entries on forms
`-t`	Multiple match option—Use when more than one entry matches. Enter a value of 3 to affect the first match, and a value of 5 to affect all matches.
`-v`	Forces override—If the user has logged in from a different IP address, this option tells the server to use the new IP address of the BMC Remedy Data Import client and invalidates the old IP address.

Option	Description
`-i`	Suppresses the default value (0) if the data being mapped to the field is empty. If the field is a non-mapped field (not present in the data file), and the value of `-i` is set to 1, then for every non-mapped field that has been defined with the default value on the Server, the import tool sets the value to NULL. However, if the field is a mapped field (even if its value is empty), the import tool does not change the value. For this, make sure that the value of `-Q` is set to 1. Note: If the value of `-D` is set to 4 (update option), then the Data Import utility does not suppress the default values while creating new records.
`-D`	Duplicate ID—Defines how to process records that contain request IDs that duplicate those already in the form. Include one of the following numbers with this option: ■ 0—Generate new ID for all records. ■ 1—Reject duplicate records. ■ 2—Generate new IDs for duplicate records. ■ 3—Replace old records with new records. ■ 4—Update old records with new records' data (the default).
`-q`	Suppresses the required field property for non-core fields. The options are `1` (on) and `0` (off).
`-Q`	If the value of this option is set to 0, the default value is set if the data being mapped to the field is empty. If you are setting the value of this option to 1, see the information given for the `-i` option above.
`-c`	Truncates character values that are longer than the field length for character fields. The options are `1` (on) and `0` (off).
`-h`	Suppresses pattern matching for fields. If supplied, the `$PATTERN$` field limit is ignored. The options are `1` (on) and `0` (off).
`-charset`	Specifies the character set used in the data file. The character set name must be supplied as listed in the IANA Charset Registry.
`-debug`	Sets the log level. The log levels are: ■ 0: OFF ■ 1: ERROR ■ 2: WARN ■ 3: INFO ■ 4: DEBUG ■ 5: TRACE ■ 6: ALL OFF does not log any information, and ALL is the maximum log level, which logs detailed log information. The default log level is 3.

To import data with a mapping file, use *either* `-M` or a combination of `-m` and `-d` to specify the mapping file to use. (You cannot use both the combination of `-m` and `-d` *with* `-M`; they are mutually exclusive.)

NOTE

The combination of `-m` and `-d` options is supported for the legacy `.arm` mapping file types only. If the mapping file is `.armx`, only `-M` is valid.

The following table describes the mapping file options. For more information, see "Importing with a mapping file" on page 255.

Option	Description
`-M`	Fully qualified path name of the mapping file to use.
`-m`	Name of the mapping file to use. You can verify the required name by opening the mapping file and using the string contained in the first line of the file.
`-d`	Directory that contains the mapping file being referenced with the `-m` option.

Using the options.xml file

The `options.xml` file, introduced in this release of BMC Remedy AR System, contains the data import commands and import parameters for single or multithreaded import. For all the data import commands included in this file, the BMC Remedy Data Import command-line utility starts as a new JVM only once.

The BMC Remedy Data Import command-line utility uses the `options.xml` file as the input. The utility identifies the location of the `options.xml` file using the new command line parameter, `-z`.

Consider the following important points before using the `options.xml` file:

- The BMC Remedy Data Import command-line utility follows the sequence of the data import commands defined in the `options.xml` file.
- Each command tag listed in the `options.xml` file will be executed. If the same command tag occurs multiple times, the BMC Remedy Data Import command-line utility executes the command tags as many number of times as listed.

IMPORTANT

The BMC Remedy Data Import command-line utility invocation command must have `-x`, `-u`, `-p`, and `-z` parameters to start importing the data files using the `options.xml` file.

- The parameters that are included in the `options.xml` file override all the parameters passed through command line.
- If any error occurs during the command execution in the `options.xml` file, the BMC Remedy Data Import command-line utility continues to execute the further commands listed in file.

- The BMC Remedy Data Import command-line utility allows sequential and parallel importing of data in a single JVM invocation instance. This is done through the `options.xml` file using the `isSerial` attribute. If the value of the `isSerial` attribute is 'False' (default) or the attribute is not specified, the BMC Remedy Data Import command-line utility imports the data by using the parallel mode. During parallel importing, the utility imports multiple data files simultaneously.
- The `options.xml` file has a global tag (optional) for global parameters. The global parameters can be overridden by individual command tags (local parameters specified in individual commands), except for the `threads` and the `debug` parameters. These global parameters cannot be overridden by local parameters.

NOTE

The `threads` and the `debug` parameters are not considered if they are specified as local parameters in a command tag format.

- If the `-o` and `-z` parameters are combined, the BMC Remedy Data Import command-line utility treats the paths specified for the `-z` parameter as relative. The tool thus combines the paths specified in the `-o` and `-z` parameters and then continues importing the files listed in `options.xml` file. If only the `-z` parameter value is specified, the path specified for the `-z` parameter is considered as the absolute path.

  For example, if the following values are specified:

  - `-o "c:\temp" -z "opt\`*`FileName`*`.arx"`

    The final path (relative path) is `"c:\temp\opt\`*`FileName`*`.arx"`

  And if the following values are specified:

  - `-z " c:\opt\`*`FileName`*`.arx"`

    The final path (absolute path) is `"c:\opt\`*`FileName`*`.arx"`

NOTE

The preceding rule is true only for the data file's path specified in the `-o` parameter; all the remaining parameters that take the file path as an input are used as absolute paths or as relative paths with respect to the current invocation directory.

- The `-z` parameter cannot be used with the `pattern` and `filelist` parameters through the command line. These parameters can only be used independently or with the `-o` parameter (as a directory).
- The data import invocation using the `-z` parameter generates a summary file containing the results of all the data import commands defined in the `options.xml` file. This summary file has the same name as the `options.xml` file. For example, if the `options.xml` file has the name, `option_fnd.xml`, the BMC Remedy Data Import command-line utility generates a summary file named `option_fnd_summary.log`.

- If the `-l` parameter (full path name of the log file) is specified for every command in the `options.xml` file, the BMC Remedy Data Import command-line utility creates separate log files for every command tag. If the log file is the same for multiple command tags in the `options.xml` file, all the logging details for these command tags are written in that one log file. If the `-l` parameter is *not* specified in the command and in the `options.xml` file, the BMC Remedy Data Import command-line utility creates a log file in the current directory with the same name as the data file name (*`datafilename`*`.log`).
- If the `debug` parameter is specified as a global parameter, the value of this parameter will be common for all the commands in the `options.xml` file.
- In the `options.xml` file, the number of threads in a pool can be configured at the global level by setting the `-threads` parameter in the global tag with the optimum value. This switch is optional; if the command does not have this switch, the value of the `-threads` parameter is set to its default value (50).

NOTE

If the `-threads` parameter is specified as a global parameter, it overrides the threads option that was provided as a command line parameter while invoking BMC Remedy Data Import command-line utility.

Sample options.xml file

The following XML tags and attributes can be used in the `options.xml` file:

- `import`—Root element of the `options.xml` file.
- `global`—Contains the global parameters with the attributes (name and value).
- `commands`—Contains the attribute, `isSerial` (default value, False) for serial and parallel importing.
- `command`—Contains the parameter with attributes (name and value).

NOTE

You can rename the `options.xml` file to any custom name. Make sure that the file contains only the above XML tags and attributes, and is a valid and well-formed file.
If the `options.xml` file is not a valid file or it does not exist, the BMC Remedy Data Import command-line utility displays an error and will not proceed further.

```
<import>
<global>
  <parameter name ="x" value="ServerName"/>
  <parameter name ="u" value="UserName"/>
  <parameter name ="p" value="Password"/>
  <parameter name ="debug" value="3"/>
  <parameter name ="threads" value="32"/>
</global>
<commands isSerial = "true">
```

```
  <command>
    <parameter name ="D" value="1"/>
    <parameter name ="o" value="DataFileDirPath"/>
  </command>
  <command>
    <parameter name ="D" value="3"/>
    <parameter name ="o" value="DataFileDirPath1"/>
    <parameter name ="e" value= "10000,10050"/>
  </command>
</commands>
<import>
```

Using the custom_options.xml file

If the source data file is in CSV or ASCII format, you can use the `custom_options.xml` file to specify date and time formats, the separators to be used, and other information.

BMC Remedy Data Import searches for formats (date and time, separators, and so on) as follows:

1. It searches the mapping file, if specified by the user.
2. In the absence of a mapping file, it searches for `custom_options.xml`, if the `-custom` command-line parameter is used.
3. If neither the mapping file nor `custom_options.xml` is specified, it searches for the formats defined by the Sun JDK for the default system locale.

NOTE

You can use the mapping file and the `custom_options.xml` file simultaneously. In such cases, the a.m. and p.m. symbol settings and the separator settings of the mapping file take precedence. However, the date and time formats in `custom_options.xml` are considered with the formats in the mapping file for parsing date and time values. In such cases, `custom_options.xml` provides additional date and time formats (the mapping file can have only one), which is helpful for parsing data files that contain date and time strings of different locales.

Importing in a multithreaded environment

To import data on multiple threads, configure the following options to specify multithreaded functionality:

- `-o`
- -z
- `-filelist`
- `-threads`

For information about these options, see "Options for BMC Remedy Data Import command-line utility" on page 245.

When running the BMC Remedy Data Import utility in a multithreaded environment, remember these tips:

- You must include a form name in the BMC Remedy Data Import utility command. (If a mapping file is provided and includes a form name, the form name is optional.)
- If any data files in the specified data directory contain data from a different form, the BMC Remedy Data Import utility cannot resolve the difference. The utility imports the data in specified form only.
- If a mapping file is specified and if the specified form uses the same mapping file for all data files, the BMC Remedy Data Import utility imports all file types (`.arx`, `.csv`, `.xml`, and `.ascii`).
- For `.arx` and `.xml` files, if you run the BMC Remedy Data Import utility without including a form name and a mapping file, data from the individual file is imported to their respective forms, so make sure that data files are not interdependent. For example, when importing Group form and User form data, users belong to groups. The User form data is dependant on Group form data, and the BMC Remedy Data Import utility cannot resolve this dependency.

  For CSV and ASCII files, you must include a form name in the command because these files do not include form information.
- The BMC Remedy Data Import utility does not validate workflow on forms where data is imported.
- For AR System 7.1.00 and later versions, the BMC Remedy Data Import utility uses bulk APIs (unless the `-e` option is used). When a bulk API fails in its first attempt, it rolls back the entire operation and retries up to the last completed entries again with bulk API. The remaining records are imported it by using individual APIs.
- Options that you specify to handle duplicate records (`-D`), bad records, and multiple matches (`-t`) are common to all of the threads.

  There is no explicit command-line option for bad-records handling. By default, the BMC Remedy Data Import utility skips the bad records.

  In an `.armx` mapping file, you can specify bad-records handling as follows:

  ```
  <datahandling badrecords="SKIP" duplicaterecords="GEN_NEW_ID"
  stripleading="false" striptrailing="false" transactionSize="0"
  truncate="false"/>
  ```

  In an `.arm` mapping file, you can specify:

  ```
  "Bad-Record-Handling: 1"
  ```

Importing with a mapping file

You can import data through the BMC Remedy Data Import utility by using a mapping file created in BMC Remedy Data Import. You can use the mapping file on any platform. For more information about BMC Remedy Data Import mapping, see the *Configuration Guide*, "Creating mapping files," page 289.

Specifying the `-M` or `-m` option in the command line determines whether you use a mapping file.

NOTE

If you are copying the mapping file between different operating systems (such as Windows to UNIX), make sure that the file is converted properly so that the operating system can read the file.

You can override specific settings saved in the mapping file by using additional options. In this way, you can use the data mappings you created for one data file and destination form for imports with a different data file and form combination.

Importing without a mapping file

Importing without a mapping refers to running the BMC Remedy Data Import command-line utility with no mapping definition to instruct the system how to map fields.

BMC recommends that you use AR Export (`.arx`) and AR XML (`.xml`) file formats when importing without a mapping file. In these file formats, field values are mapped by matching field IDs, which are both included in the file. For CSV files, field values are mapped by matching the field labels (if present in the file) to the field names (database names of the fields) retrieved from the server. When a browser or BMC Remedy User exports data into a CSV file, the field labels and the field names are not necessarily the same. Only fields with names and labels that match are auto-mapped. Consequently, if CSV files do not include field labels, the field values cannot be mapped.

Without a mapping file, include the server name, form name, and data file name in the command line. Mappings are built by querying the server and the data file.

Localization tips

On Chinese UNIX systems, CSV and ASCII files cannot be imported if they contain Date/Time fields.

When using the BMC Remedy Data Import utility on Japanese UNIX systems, convert the data and `.arm` mapping files to EUC format before the files are moved to the UNIX server. (The `.arx` and `.xml` data files are already in EUC format when they are generated in the client tool, but the `.csv` file is not. Therefore, the `.arm` and `.csv` files must be converted.) Make sure that all of the data file names and a mapping file names are in English.

BMC Remedy Data Import utility examples

In multithreaded environments

The following examples show you how you can use the BMC Remedy Data Import utility in a multithreaded environment:

- In the following example, the `-o` option specifies the path to the directory that contains the data files to import. All of the data in the files is imported to the specified form.

  ```
  com.bmc.arsys.apiext.data.DataImport -x serverName -u userName
  -p password -f formName -o dataFilePath -l logFilePath
  ```

  For example:

  ```
  com.bmc.arsys.apiext.data.DataImport -x machine1 -u joeuser
  -p 1a2b3c -f HelpDesk -o "C:\files" -l "C:\files\test.log"
  ```

  The data in the files in `C:\files` is imported to the HelpDesk form. A log is created in `test.log`.

- In the following example, the `-filelist` option is used with `-o`, and only the data files listed are used. All of the data in the files is imported to the specified form.

  ```
  com.bmc.arsys.apiext.data.DataImport -x serverName -u userName
  -p password -f formName -o dataFilePath -filelist listOfFiles
  -l logFilePath
  ```

  For example:

  ```
  com.bmc.arsys.apiext.data.DataImport -x machine1 -u joeuser
  -p 1a2b3c -f HelpDesk -o "c:\files" -filelist "user.arx,
  user.csv,abc.xml,xyz.ascii" -l "c:\files\test.log"
  ```

  The data in the `user.arx`, `user.csv`, `abc.xml`, and `xyz.ascii` files in `C:\files` is imported to the HelpDesk form. A log is created in `test.log`.

With a mapping file

The following examples show you how you can use BMC Remedy Data Import utility with a mapping file:

- In the following example, the server name, form name, and data file name are optional because the mapping file contains the information:

  ```
  com.bmc.arsys.apiext.data.DataImport -u userName -p password
  -m mappingFileName -d mappingFileDir -l logFile
  ```

- In the following example, the server name, form name, and data file name override the names in the mapping file. When you use the BMC Remedy Data Import utility with a mapping file, you can override one or more of those names.

  ```
  com.bmc.arsys.apiext.data.DataImport -x serverName -u userName
  -p password -m mappingFileName -d mappingFileDir -l logFile
  -o dataFilePath -f formName
  ```

Without a mapping file

Without a mapping file, you must specify the server name and data file name because there is no mapping file to provide such information.

The `-d` and `-a` options are not shown in the following examples, but if you work with multiple servers on the same computer, you can use `-d` for duplicate record handling and `-a` to specify a port number.

The following examples show how you can use the BMC Remedy Data Import utility without a mapping file:

- In the following example, minimal options are used. The *`dataFilePath`* specifies the data file with path to import. If there are multiple data sets in the same data file, an import is attempted for *all* forms.

  ```
  com.bmc.arsys.apiext.data.DataImport -x serverName -u userName
  -p password -o dataFilePath -l logFile
  ```

- In the following example, the *`formName`* determines which set of data from the data file is imported to the server. The form name on the server and in the data file must match.

  ```
  com.bmc.arsys.apiext.data.DataImport -x serverName -u userName
  -p password -f formName -o dataFilePath -l logFileName
  ```

- In the following example, an import is being attempted into the form called formA on the server, but the data comes from formB in the data file.

  ```
  com.bmc.arsys.apiext.data.DataImport -x serverName -u userName
  -p password -f "formA=formB" -o dataFilePath -l logFileName
  ```

Using the BMC Remedy User CLI

Some legacy integrations might require starting BMC Remedy User from the command line. To start BMC Remedy User from the command line, use the `aruser` command.

This command takes the following parameters:

- `app`—The AR System application to launch BMC Remedy User in application

mode

- `auth`—The external authentication string
- `user`—The BMC Remedy User user name
- `password`—The password for the user

For example, to start BMC Remedy User and automatically log in with the user name “Demo” and password “remedy,” use this syntax:

```
Aruser /user="Demo" /password="remedy"
```

Or

```
Aruser -user="Demo" -password="remedy"
```

To use an external authentication string, include the `auth` parameter:

```
Aruser /user="Demo" /password="remedy" /
auth="authenticationString"
```

Or

```
Aruser -user="Demo" -password="remedy" -
auth="authenticationString"
```

NOTE

To determine what servers to log into, BMC Remedy User checks the current `AR` file. For information about how to modify this file, see the *Configuration Guide* , *"ar," page 348*.

Chapter 19 Running external processes (Run Process)

Run Process actions can be used for integration on both the AR System clients and servers.

The following topics are provided:

- Overview (page 260)
- Client and server processes (page 260)
- Using Run Process to start applications (page 260)
- Using Run Process / $PROCESS$ to retrieve data from applications (page 264)
- Run a process on the web (page 266)
- Issues and considerations (page 267)

Overview

One of the simplest ways to integrate two applications is to execute one application from within another. AR System enables you to include execution of external applications as part of workflow to enhance or supplement the features of AR System.

The reverse case, where another application executes an AR System client, is also valid. See Chapter 18, "Exporting and importing data and definitions."

Beyond simply starting the external application, AR System provides process-control functionality for these types of integration:

- **Data passing and retrieving**—When AR System executes external applications (either manually or automatically), information from any form in the AR System database can be extracted and passed as run-time arguments. You can also retrieve data by using a Run Process command and place it in a field.
- **Client and server execution**—External applications can be executed locally on the AR System client, or remotely on the AR System server.
- **Synchronously and Asychronously**—Run Process on a filter and escalation is asynchronous. All other Run Process commands (including `$PROCESS$` in a Set Fields action) run synchronously.

Executing an external process is done using the Run Process workflow action, available for filters, active links, and escalations, or in a Set Fields action with the `$PROCESS$` keyword.

Additional information is available in the *Workflow Objects Guide*.

Client and server processes

The Run Process action can be triggered on both AR System clients and servers. This provides more implementation options than the DDE or OLE Automation interfaces, which are available on the clients only.

All three workflow components can run processes to provide centralized integration on the server. In addition, active link processes can provide local integration on the clients.

Using Run Process to start applications

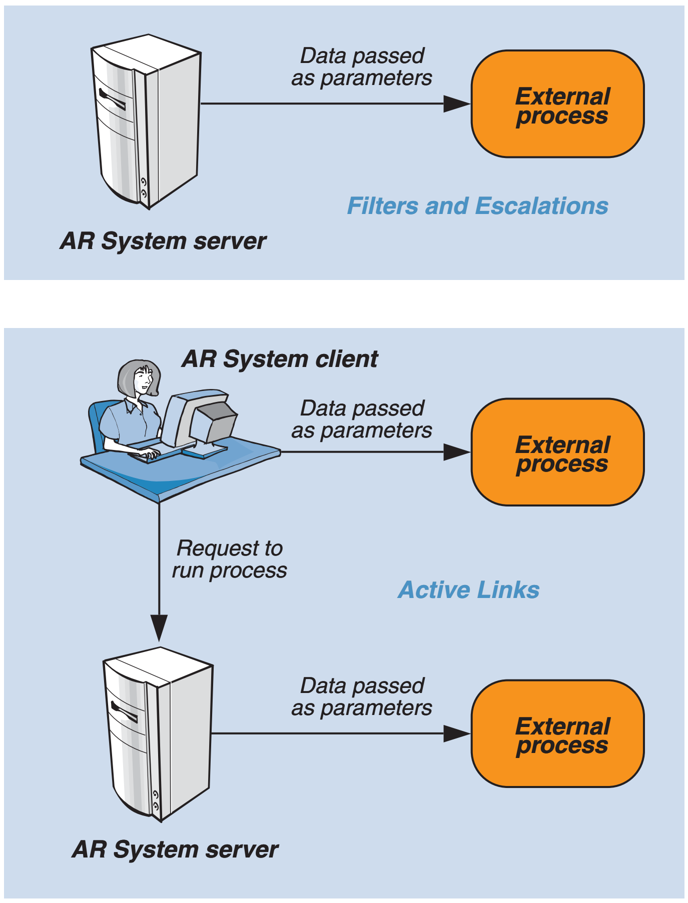
The Run Process action starts an external application. Depending on the function and behavior of the application, it can be started through any of these means:

- By a user (from an active link)
- Automatically under certain states (from a filter)
- Automatically under certain time conditions (from an escalation)

For example, a paging program can be called whenever a record marked Urgent is entered into the database (a filter action), or when such a record has not been accessed for two days (an escalation action). When the application is started, data from the current form can be passed as run-time arguments to the application.

The Run Process action simply executes an independent process; it does not return a value to the calling program.

Figure 19-1: Executing another application



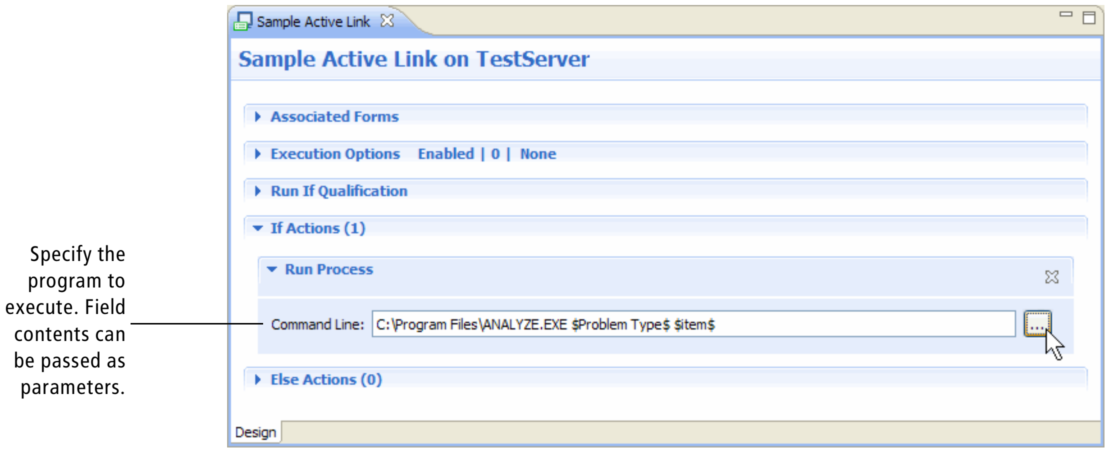
Unlike active links, which run with the access control permissions of the user, filters and escalations run with the permissions of the administrator and thus have access to all form fields. Consider this when you define filter or escalation actions because they can have security implications.

On a Windows server, a filter or escalation can run only processes that run in a console (like a `.bat` script) or that create their own windows.

The processes that an active link launches can run on the local client machine or the server. They are often triggered by actions taken by the user. For example, an external email program could be started whenever the user clicks a button on an AR System form, or a problem resolution tool can be invoked when a problem description is entered into a field.

An example of a Run Process action definition for an active link is shown in Figure 19-2. When the active link is triggered, the system executes the command specified in the Command Line field, which launches a related process. To make sure that the executable is correctly executed on the client machine, specify its full path name. When the application is started, data from the current form can be passed as run-time arguments to the application.

Figure 19-2: Defining a Run Process action for an active link


Sample Active Link
Sample Active Link on TestServer
Associated Forms
Execution Options Enabled | 0 | None
Run If Qualification
If Actions (1)
Run Process
Else Actions (0)
Design
Specify the program to execute. Field contents can be passed as parameters.

Command Line: C:\Program Files\ANALYZE.EXE $Problem Type$ $item$

When designing an active link that uses a Run Process action on the client, always consider the variety of client platforms that users will use. Keywords can be used in the Run If expression for the active link to verify that the client is on an appropriate platform. For example, the `$CLIENT-TYPE$` keyword identifies whether the client is BMC Remedy User. For more detail about the possible values for `$CLIENT-TYPE$`, see the `AR_CLIENT_TYPE` constants defined in *`ARSystemServerInstallDir`*`\api\include\ar.h`. If the active link is to be supported on multiple platforms, each platform might require its own active link with an appropriate qualification.

Example: open a reference document from an active link button

From your help desk application, you want to open reference documents that are in Word format.

1 Create a form named Ref Docs that has two fields:

- **Document Name**—Contains the name of a reference document.
- **Doc Location**—Contains the path to where the document is stored.

2 Enter data for all of your documents into this form.

3 On the appropriate form of your help desk application, add two fields:

- **Reference Documents**—A *visible* field that has a Search Menu attached that queries the Ref Docs form to build a menu of all of the Document Name titles.
- **Path**—A *hidden* field that is filled in using a Set Fields active link with the corresponding Doc Location path whenever a Document Name is selected from the menu on the Reference Documents field.

4 On the same help desk application form, create an active link triggered by a button labeled Open Document.

The qualification on the active link is that the Reference Documents field is not empty. The active link action is to do a Run Process with a Command Line specification of something like:

```
C:\program~1\micros~1\office\winword.exe $Path$
```

When a reference document is selected from the menu and the active link button clicked, Word starts, and the selected document appears.

Example: call a pager application from a filter

When a service request is submitted or modified with a severity of Critical, you want to send a pager message to the person identified in the Responsible Person field on the request. You use a pager application called TelAlert.

You need a filter that has the following characteristics:

- Executes on Submit or Modify
- Runs if

  ```
  'TR.Severity' = "Critical" AND 'DB.Severity' != "Critical" AND
  'Responsible Person' != $NULL$
  ```

  In other words, it runs whenever a trouble report is set to Critical for the first time and the Responsible Person field is not blank.
- Sends a pager message to `$Responsible Person$`.

You would use the following command for the Run Process filter:

```
/usr/telalert/telalertc -c PageMart -PIN $Pager Access Number$
-m "Trouble Report $Call ID$ has just been set to Severity =
Critical."
```

This command performs the following actions:

- It calls the TelAlert application. It uses the `telalertc` executable, which is the standard TelAlert client, instead of the `telalert` executable, which is the client plus the administration function.
- The `-c` parameter tells TelAlert to use the `PageMart` configuration information in the `telalert.ini` file.
- The `-PIN` parameter takes the value of the field Pager Access Number and passes it to PageMart to identify the specific pager that should receive the message.
- The `-m` parameter specifies the message that is to be sent to the pager. The value of the Call ID field is substituted into the message text.

Using Run Process/$PROCESS$ to retrieve data from applications

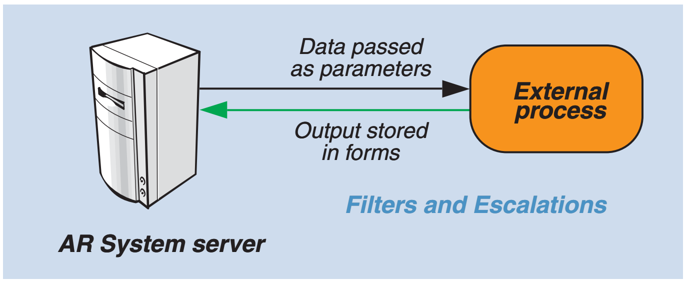
Another type of action, Set Fields, enables you to include workflow that automatically sets the contents of fields from a variety of data sources.

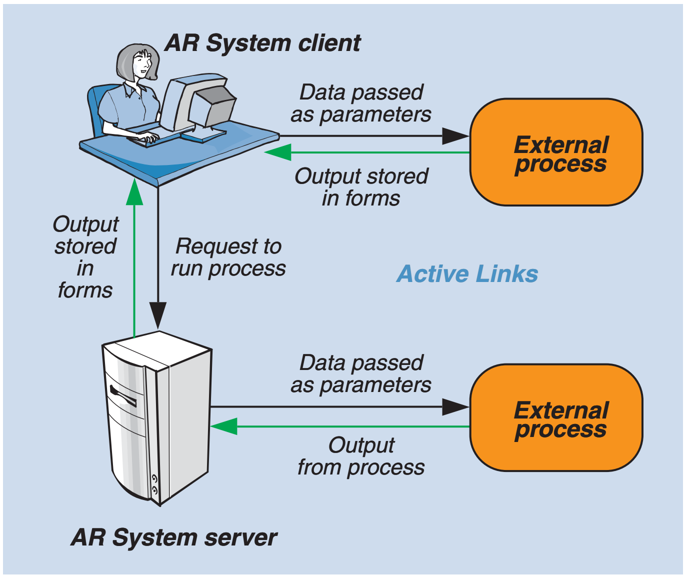
These data sources include fixed values, values read from other forms (possibly in other databases), values from arithmetic operations or functions, and values returned by external applications. Using the `$PROCESS$` keyword of the Set Fields action, AR System can execute an application and set the output of the application in various data fields. You can run only a process that behaves as follows:

- Runs in a console (such as a `.bat` script or `runmacro.exe`), but not in a GUI application.
- Returns 0 if successful. (In this case, `stdout` is pushed to the field.) If the process returns anything other than 0, `stdout` displays an error message.

Whether you use an active link, filter, or escalation depends on the purpose of the external application. Active links can execute locally on the client machines or on the server. Filters and escalations execute only on the server.

Figure 19-3: Retrieving data from another application





When an external application is run on the server, AR System waits for the process to terminate so that it can capture and interpret the output of the process. To avoid situations where AR System waits indefinitely for a process that fails to terminate, a process time-out is built into AR System. This time-out can be configured for between 1 and 60 seconds, using the AR System Administration: Server Information form.

In a Set Fields definition, the keyword `$PROCESS$` indicates that all following text is a command. Use the full path name to the executable. AR System data field values can be passed as parameters. When using active links, remember that they run with the access control of the user, so access to form fields might be limited.

When workflow that performs a Set Fields action is fired, the process starts, and BMC Remedy User waits for it to complete. (In UNIX, the process runs in a Bourne shell.) All returned data is read by BMC Remedy User and processed according to the return status of the process:

- If the return status is zero, the data is used as the new value for the field.
- If the process returns with a status other than zero, BMC Remedy User assumes the process failed and does not update the field contents. Instead, the output from the process is used as an error message and displayed to the user.

When designing an active link that uses a `$PROCESS$` Set Fields action on the client, always consider the variety of client platforms that users will use. The keywords `$HARDWARE$` and `$OS$` can be used in the Run If expression for the active link to verify that the client is on an appropriate platform. If the active link is supported on multiple platforms, each platform requires its own active link with an appropriate qualification.

NOTE

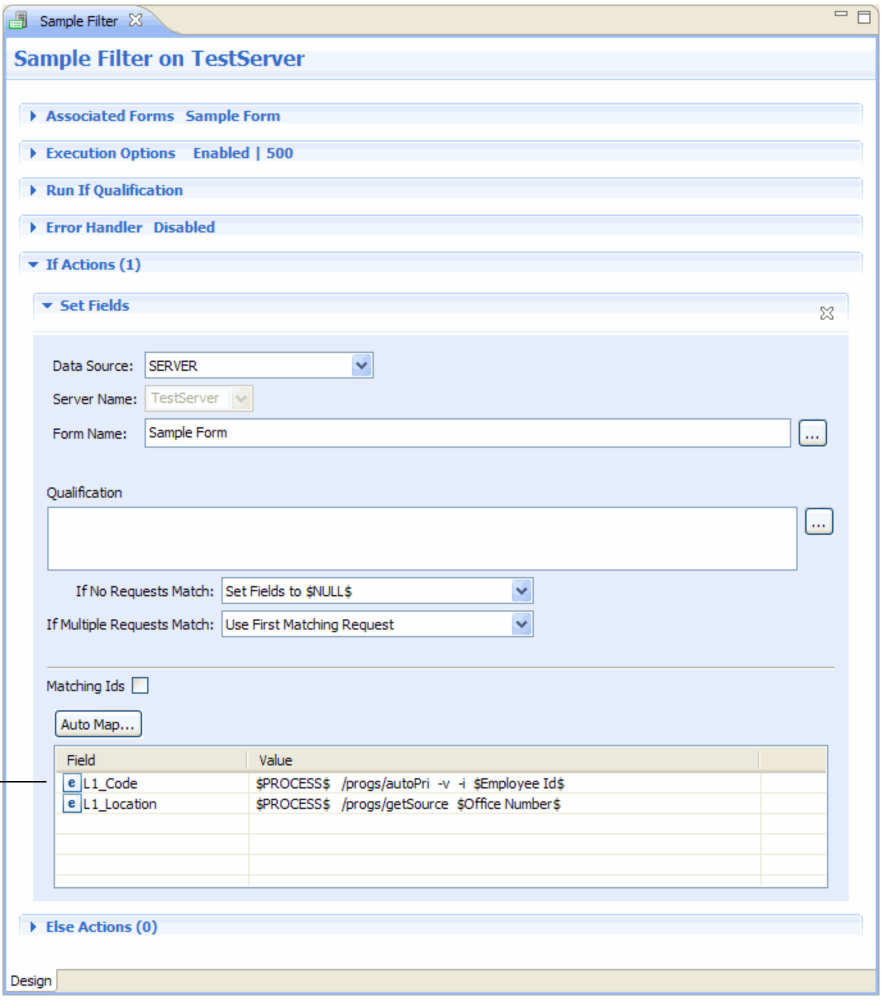
You can run a process on a server by inserting `@`*`serverName`*`:` before the process name in an active link. For example,

```
$PROCESS$ @ServerA: C:/temp/process.exe
```

If it is the current server, you can use @@ instead of `@`*`serverName`*.

An example of a Set Fields action for a filter is shown in Figure 19-4. In this example, the action sets the values of two fields by executing a separate utility program for each one, passing values of existing fields as parameters. If the programs execute correctly (that is, return with an exit code of zero), their outputs are assigned to the respective fields.

Figure 19-4: Defining a Set Fields action using $PROCESS$



Sample Filter on TestServer

Associated Forms Sample Form

Execution Options Enabled | 500

Error Handler Disabled

Set Fields

Data Source: SERVER

Server Name: TestServer

Form Name: Sample Form

Qualification

If No Requests Match: Set Fields to $NULL$

If Multiple Requests Match: Use First Matching Request

Matching Ids ☐

Field	Value
L1_Code	$PROCESS$ /progs/autoPri -v -i $Employee Id$
L1_Location	$PROCESS$ /progs/getSource $Office Number$

The value of the L1_Code field is set by an external program called autoPri, which accepts an employee ID as an argument.

Run a process on the web

JavaScript is an HTML scripting language that allows you to create programs that reside directly on an HTML page. An active link can use the Run Process action to run JavaScript on the browser. The JavaScript code must have the word `javascript` in front if it. For example, the following code shows an alert box with “Hello world” in it:

```
javascript alert("Hello world");
```

You can use keywords and field references in the JavaScript, for example:

```
javascript alert("You are in $SCHEMA$ and Submitter is $2$");
```

In several BMC Remedy applications, the user is shown a table of related tickets along with the primary ticket. These related tickets can be from different forms. A special form is maintained, which records relationships between tickets. The related tickets table field shows this special form. When the user double-clicks on any of the related tickets, instead of showing the special form to the user, an open window action opens the form that has the ticket data.

Limitations in using JavaScript

- All the Run Process JavaScript actions are grouped together and executed at the end of the active link. For example, if you have a Run Process action followed by a Set Fields action, the Set Fields action is executed before the Run Process action.
- Some JavaScript code is asynchronous. For example, openModifyForm starts the process of opening the form, but does not wait for the action to complete. So it is not possible to have another Run Process action that clicks a button on the newly opened form.
- Any special characters in JavaScript must be properly escaped. For example, if the action has JavaScript `alert("Short Description is $8$")` and the Short Description value contains a double quote or a backslash or a new line, a JavaScript error occurs.
- If the word `javascript` is not at the beginning of a run process action, it says “`The following specific error(s) occurred when executing ‘xx’,`” but it does not say what the error is.

Issues and considerations

- Active links that run a process on the client should have qualifications that limit usage to an appropriate client platform. The keyword `$HARDWARE$` can be used to check for the client platform. On UNIX machines, `$HARDWARE$` returns the value of the command `uname -m`. On the Windows clients, it returns the value `PC`.
- On UNIX machines, processes run under the Bourne shell.
- When passing data fields as parameters to external programs, enclose the arguments with double quotation marks if the value might contain spaces or special characters.
- The `$PROCESS$` feature of the Set Fields action is effective for dynamically pulling or loading small amounts of data on the AR System client or server. For large amounts of data, use the API.
- The Run Process string can have a maximum length of 4096 bytes. To execute large scripts, use field references to build the script’s data.
- For information about special Run Process commands, see the *Workflow Objects Guide*.

Chapter 20 OLE automation

You can use the OLE automation active link action to integrate BMC Remedy User with an external automation server. The OLE automation action is just like any other active link action—when it executes, it carries out the specified automation sequence on the specified automation server.

The following topics are provided:

- OLE overview (page 270)
- AR System and OLE automation (page 271)
- Active links and OLE automation (page 272)
- Maintaining server context across multiple active link actions (page 278)
- Working with ActiveX controls (page 278)
- AR System as an OLE automation server (page 278)
- DCOM support (page 279)
- The OLE automation active link action (page 279)
- Issues and considerations (page 284)

For more information about automation in AR System, see the *C API Reference*.

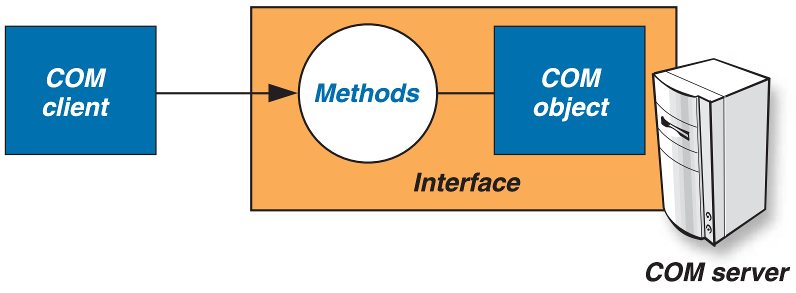
OLE overview

OLE (Object Linking and Embedding) is Microsoft technology that allows one application to send commands or data to another application.

The terminology in this area is often confusing, but OLE Automation is the only aspect of OLE that AR System addresses. First, however, you should understand the Component Object Model (COM), the framework for developing and supporting component objects. Using COM, many different types of software can interact in a predictable fashion.

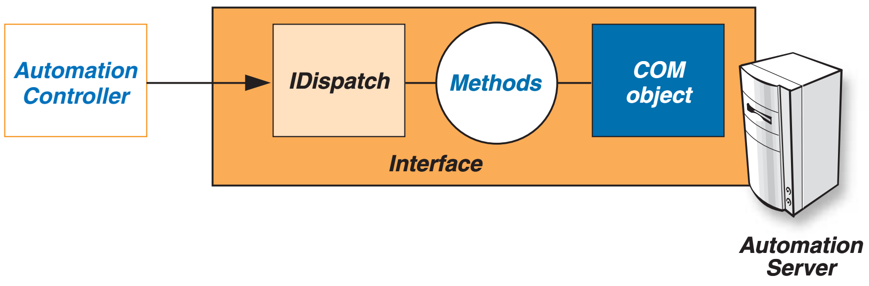
A *COM object* is software that provides its services to other software through one or more *interfaces*, each of which includes several methods. A *method* is typically a function or procedure that performs a specific action and can be called by the software that uses the COM object (the *client* of that object).

Figure 20-1: Generic COM architecture



OLE Automation, sometimes referred to simply as Automation, is a means for a COM client to control a COM object (or component). An automation *server* is a COM object that implements the `IDispatch` interface, which is a predefined COM interface. An automation *controller* is a COM client that communicates with the automation server through that interface.

Figure 20-2: OLE Automation architecture



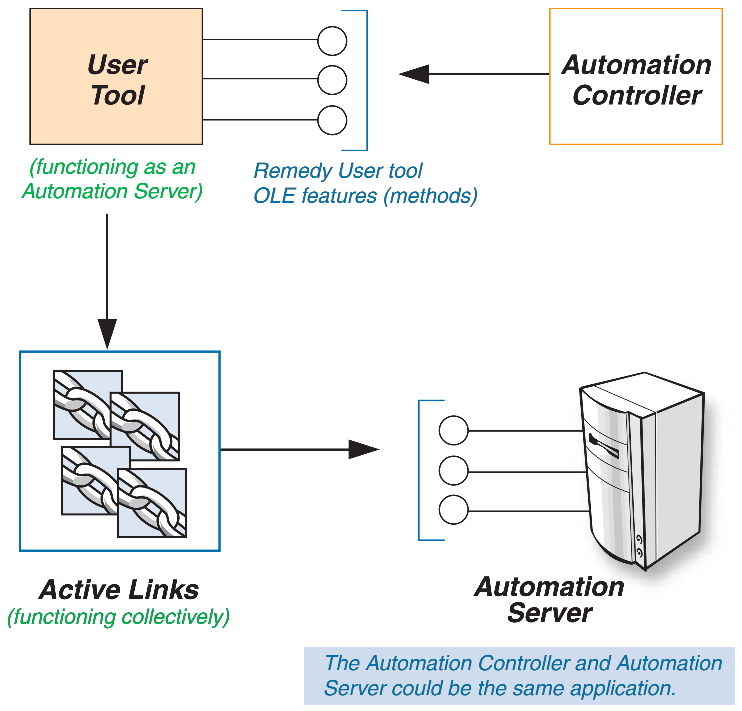
When Microsoft added Web-aware features to OLE, it changed the name to ActiveX. An *ActiveX control* is an example of an Automation server that can be controlled by BMC Remedy User. ActiveX controls are objects that can be reused by many applications; many of them can be downloaded as small programs or animations for Web pages.

AR System and OLE automation

BMC Remedy User can be an Automation server or an Automation controller. As an Automation controller, it uses active link actions to send commands to other applications. For example, you can enable your forms to use electronic phone pads, spreadsheets, charts, graphs, spell checkers, or employee pictures. You can also write OLE Automation servers that are specific to your needs.

As an Automation server, BMC Remedy User can be accessed by another application. The other application can perform such functions as opening forms, creating entries, opening guides, or running macros. The following figure shows how a complete cycle of Automation could be designed. In the figure, `IDispatch` is not shown, but it implicitly mediates between Automation controllers and servers. In essence, the brackets symbolize `IDispatch`.

Figure 20-3: OLE Automation in AR System


User Tool
(functioning as an Automation Server)
Remedy User tool OLE features (methods)
Automation Controller
Active Links
(functioning collectively)
Automation Server
The Automation Controller and Automation Server could be the same application.

Active links and OLE automation

In AR System, active links allow access to Automation servers through the methods and properties that are accessible in its type library. This section discusses the servers, the type libraries, and the specific active link interface.

Automation in AR System supports two types of servers:

- **Local servers**—Executable OLE servers that run in their own process space.
- **In-process servers and controls**—Dynamic link libraries (DLLs) that run in the same process space of the invoking process.

Automation servers, controls, and type libraries are defined in the system registry. BMC Remedy Developer Studio reads this information and displays this information when the user defines any Automation active link. AR System uses the following rules when it reads the system registry.

- The Automation server must have the following properties listed under its registry entry:
  - `TypeLib` listing the typelib IID (interface identifer) or `Programmable`.
  - `InProcServer` or `LocalServer` entry.
- When the servers have any one of the following properties, they are not displayed in BMC Remedy Developer Studio:
  - If the in-process server (`InProc`) is a system component, not an application OLE server, it is skipped.
  - If the Automation server is an OLE 1.0 control, it is skipped. (All 1.0 OLE controls have an `Ole1Class` entry.)
  - If the server entry has the `autoconvert` attribute, that server entry is ignored.
- An Automation system component has the following registry entries for InProc servers under the `InProcServer32` entry:
  - `ole32.dll`
  - `oleprx32.dll`
  - `ole2pr32.dll`
  - `olecnv32.dll`
  - `oleaut32.dll`
- An Automation system component has the following registry entries for `InProc` server under the `InProcServer` entry:
  - `ole2prox.dll`
  - `ole2.dll`
  - `ole2disp.dll`

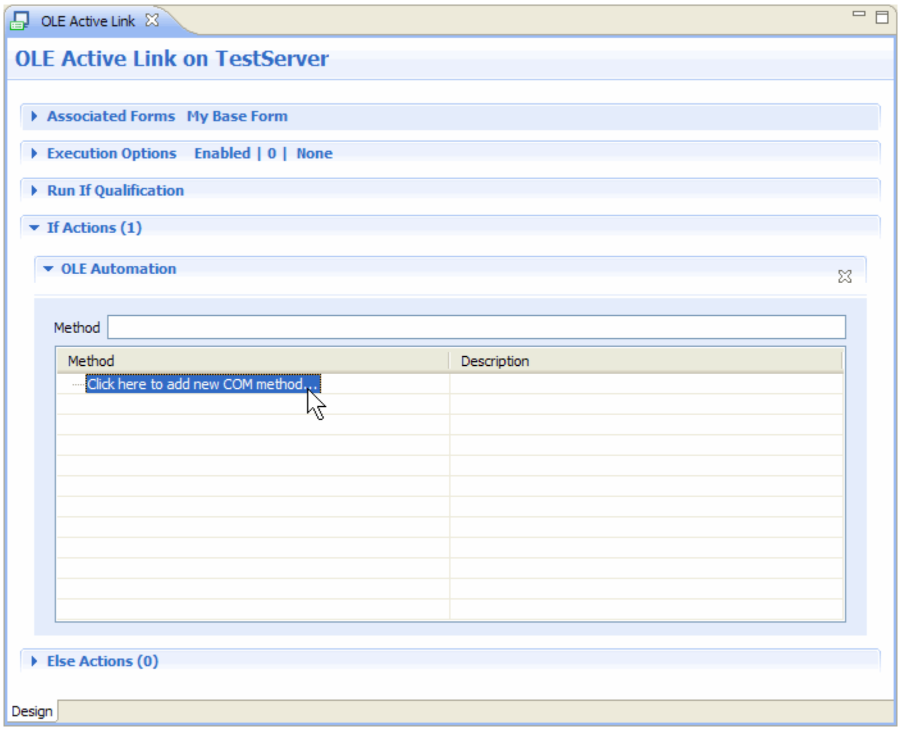
Using the GUID

Active links defined using an Automation server or control must use the globally unique identifier (GUID) of that server or control. A GUID is a unique 128-bit number produced by Windows (and some Windows applications) to identify a particular component, application, file, database entry, or user. Therefore, each user machine where an active link executes must contain the same server or control where the active link was defined. If you build your own Automation server or control, you must register it on the user machine using the `Regsvr32` utility. For more information about registering servers and controls, see the Microsoft OLE, COM, and ActiveX documentation.

The OLE automation active link interface

You must understand how objects, methods, parameters, and return values are identified in the If Action or Else Action panels for creating OLE Automation active links.

Figure 20-4: OLE Automation active link action


OLE Active Link
OLE Active Link on TestServer
Associated Forms My Base Form
Execution Options Enabled | 0 | None
Run If Qualification
If Actions (1)
OLE Automation
Method
Method
Description
Click here to add new COM method...
Else Actions (0)
Design

NOTE

The OLE Automation active link user interface is not a full development environment. For example, debugging tools are not available. The best way to develop the sequence of method calls is to use a product such as Visual Basic. With an operational Visual Basic program, you can enter method calls by using the procedures described in this section.

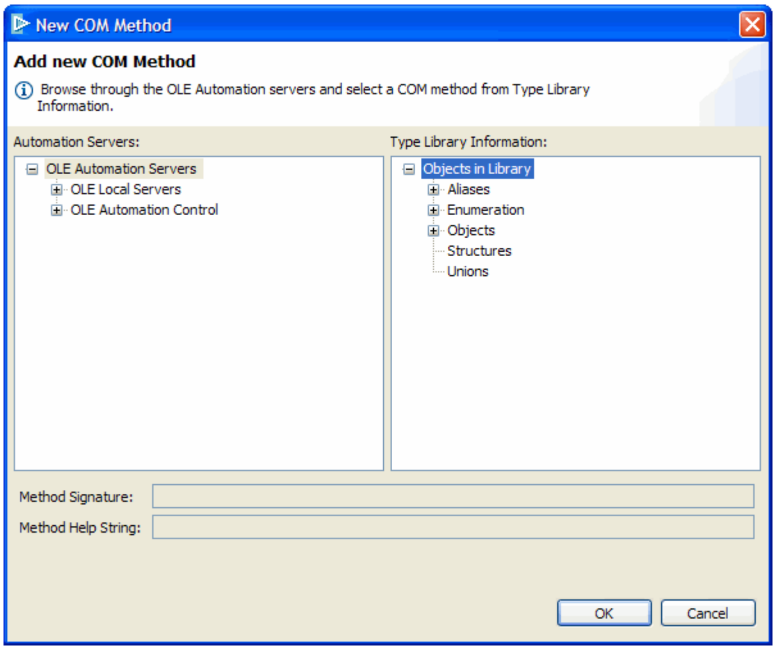
You can also obtain documentation for working Visual Basic programs and choose not to develop them yourself. However, Customer Support cannot help you develop Visual Basic programs.

When a particular AR System server is selected, BMC Remedy Developer Studio queries the type library information from the registry and displays the information in a tree format in the Type Library Information box. The Type Library tree displays the following data types:

- **Aliases, enumerators, structures, and unions**—Special data types defined by the Automation authors and can be used as parameters or return types in the method calls defined in the objects. These data types are used for display purposes only.
- **Objects**—The focal point of the Automation system. Objects contain methods and properties. The object node displays all of the objects defined by the Automation server author. Under each method node, all methods are displayed in a C style syntax:

  `*returnType methodName* (parameters)`

Figure 20-5: OLE Automation active link action—type library


New COM Method

Add new COM Method

Browse through the OLE Automation servers and select a COM method from Type Library Information.

Automation Servers:
- OLE Automation Servers
  - OLE Local Servers
  - OLE Automation Control

Type Library Information:
- Objects in Library
  - Aliases
  - Enumeration
  - Objects
  - Structures
  - Unions

Method Signature:

Method Help String:

AR System does not show object methods (which have safearrays and c-arrays as return-types or as parameters) because these types are not supported.

If a `Get/Set` accessor type is defined, BMC Remedy Developer Studio displays the object properties as methods. For example, a `BSTR MyString` property is displayed as two methods:

- `BSTR MyString()`
- `Void MyString(BSTR rhs)`

These two methods can be used to `Set` and `Get` the MyString property values.

Reading the method tree

After you chose a server, an object, and a method to invoke, the method is added to the call sequence. A method tree and associated parameters are displayed in tabular format on the OLE Automation panel. The method node along with return types are also displayed. If the return type of a method is void, no return type node is displayed. The return type can be assigned to a field type, nested to operate on the next method, or be substituted for a method parameter type.

The following example shows how to open Excel by using three actions in an active link:

1 Make automation server controllable by the user interface:

```
_Workbook:Application._Application:User Control(1)
```

NOTE

If `user control` is not specified, Excel terminates when the automation sequence ends. If you plan a series of sequential Automation actions, `User Control` must be invoked.

2 Make Excel visible:

```
_Workbook:Application._Application:Visible(1)
```

3 Open the workbook:

```
_Workbook:Application._Application:Workbooks.Workbooks:Open()
```

Automation allows nesting only one level deep when building method sequences. A typical sequence might be as follows:

```
object1().object2()
```

```
object1().object2(object3())
```

This sequence requires three trips through `IDispatch`.

Working with method parameters

Parameter values can be supplied as follows:

- Parameters can be filled in with the return value of another method.

  When you nest, the return type of the method to add and the parameter type should match. BMC Remedy Developer Studio validates the type compatibility when you nest the methods.

- You can fill in parameters. The user-defined parameters are validated as follows:

  - From the method tree, all parameters are extracted as strings including integer, double, and so forth.

  - A variant change to the native OLE type is then performed to validate the type of the parameter the user has entered. If the variant change failed, BMC Remedy Developer Studio reverts to the previous state, ignores the latest change, and displays an error message indicating a data type mismatch has occurred.

  For example, assume that in a method called `SetInt()`, the user has entered `1234` as the parameter value. As soon as the focus moves off of the parameter value node, denoting completion of the action, BMC Remedy Developer Studio extracts `1234` as the string `1234` and tries to convert it to an integer. If the change is possible, the user's changes are accepted; otherwise, an error message is displayed. If the user entered `Test` as the parameter value, the variant change operation would fail with the appropriate error message displayed.

  If a parameter is a pointer type variable and the user types in a value, BMC Remedy Developer Studio displays an error message. In this case, the user is allowed to add only methods that have the same pointer type as the return value.

  A parameter that is an object pointer does not accept a typed-in value but accepts a method supported in the ActiveX control that returns that object.

  Methods of other interfaces can be selected, but make sure the return type of the method to add and the parameter type are the same.

  A right mouse click on the parameter value node displays a menu to select AR System Form field values or keywords. No validation is done when field values are assigned to parameters.

  No type validation is done on the `VARIANT` parameter type. BMC Remedy Developer Studio cannot determine the underlying type of the variant at design time. However, BMC Remedy User performs validation at run time and displays any error messages if a type mismatch occurs.

Working with method return types

Values cannot be entered directly at the return node. However, AR System fields can accept the return value of the method. Users can click the right mouse button to display a menu list, and users can select field names from the form. Keywords also appear in the menu list but are grayed out because return values for a method call cannot be assigned to keywords, only to variables. If the menu list shows grayed-out keywords and fields, a return value cannot be assigned because of a type incompatibility. This is usually the case with pointer types.

Other methods can be nested to act upon the return type. For example, to make Word visible, the following method call is used:

1 Select the Word Document local server.

2 Select the `_Document Object` and the `Application*Application()` method. This method returns an application object.

3 Select the `_Application Object` and the `Void Visible` method. Any method can be nested from the application object.

When a method is added to the return type, BMC Remedy Developer Studio uses the following rules to validate the method:

- BMC Remedy Developer Studio checks to determine if the return type of the last method in the call sequence is of type Object. An error message is displayed if the return type is not an object.
- If the return type is an object and another method is added to the return type, the Interface Identifier (IID) of the returned object and the IID of the method being added are compared. The method is validated if the user is adding a method from an object, which is compatible to the return type of the last method in the call sequence.
- If the return type of the last method in the call sequence is of type `IDispatch`, IID validation is not performed because BMC Remedy Developer Studio cannot predict which object the `IDispatch` interface represents at design time. (BMC Remedy Developer Studio validates only that a value cannot be directly entered like an ordinary parameter.)
- If the return type is `VARIANT`, no validation is performed because BMC Remedy Developer Studio does not know the underlying type at design time. For example, if the object `Ifoo` supports two methods, `VARIANT GetBSTR()` and `SetDispatch(VARIANT)`, BMC Remedy Developer Studio allows the nesting of methods as `SetDispatch(GetBSTR())` because BMC Remedy Developer Studio is not aware that the `VARIANT` from `GetBSTR` contains a `BSTR` value and not an `IDispatch` value. BMC Remedy User performs a validation at run time, and an error message appears if a type mismatch occurs.

Maintaining server context across multiple active link actions

AR System automatically maintains the OLE server context across multiple active link actions. The server's context is maintained across each active link action so that subsequent actions act on the same server. When the active link finishes firing, the server's context is released. However, depending on the local server's implementation, the executable might close if all active connections to the server are terminated.

Working with ActiveX controls

The OLE Automation active link is open to Visual Basic and other languages. OLE Automation active links can be constructed to invoke Visual Basic ActiveX controls from AR System. After your callable ActiveX control is built, make sure that the class module and functions are public and that a recognizable name is given to the interface being exposed.

Follow these rules when creating an ActiveX `.dll` in Visual Basic:

- Provide a recognizable name to the interface being exposed. Each defined class module becomes an interface object. In Visual Basic, select the Class property and change the ClassName.
- Provide a recognizable name to the Project Name. In Visual Basic, choose Project Menu > Properties > ProjectName.
- Make sure the DLL is created and registered in the system by using the `Regsvr32` utility, for example, `regsvr32` *`dllName`*.

When you use an ActiveX control as an OLE server, you select an ActiveX control from the list of OLE Automation Controls, rather than the list of local OLE servers.

AR System as an OLE automation server

BMC Remedy User exposes its basic functionality to OLE Automation clients located on the same machine. This functionality includes the ability to open a form, perform create, search, and modify operations, get and set field values and properties, open a guide, and run a macro. For more information, see the *C API Reference*.

You can manipulate BMC Remedy User directly or create a Visual Basic control and manipulate the Visual Basic DLL through the Automation active link as described in the preceding sections.

Your client application can launch a new instance of BMC Remedy User or connect to an instance that is already running. It can also connect to a running AR System application server object (an instance of BMC Remedy User that manages a prescribed set of forms), but it cannot launch a server object.

Unless otherwise specified for an interface, your client application can both get and set the interface object's properties. As with the AR System API, the AR System permissions you have when making an OLE Automation call are those of the user logged in to the session you use.

The following example illustrates the use of BMC Remedy User through the exposed inbound interface to invoke the an AR System form through an Automation active link:

1. Select `BMC Remedy User Application Class` (the BMC Remedy User inbound server name) from the LocalServer node.
2. Select the `IcomApp` object and then select `OpenForm()`.
3. Provide the parameter values. The OpenForm call returns an `IschemaWnd` object. For example, call `GetServerName` as a nested call to obtain the server name. Return values can also be assigned to an AR System form. The call structure is `$field$ = OpenForm().GetServerName()`.

DCOM support

Distributed COM (DCOM) is only partially supported by AR System. All Automation servers invoked by AR System are treated as local servers and not remote DCOM servers. To use the DCOM protocol, you can create a local server that calls an outside DCOM process. The same Automation servers must be present on both the AR System administrator's machine and the AR System users' machines; otherwise, an error message is displayed indicating that the servers are not available.

The OLE automation active link action

Use the OLE Automation action to share functionality between applications that support OLE. When using this action, AR System acts as an OLE automation controller client to an OLE server.

The following procedure describes how to implement OLE automation in AR System. As in SQL, you should debug OLE automation calls in a full OLE and COM development environment and then use that knowledge to build the same calls in BMC Remedy Developer Studio.

A sample exercise is outlined in "OLE automation example—Sample:SpellCheck" on page 283.

NOTE

If you design an active link for a form that is also used by clients on platforms other than Windows, verify that the current platform is a PC as a condition of activating the OLE action. To do this, include a qualification that uses the `$HARDWARE$` keyword to verify the current platform. For more information about building qualifications, see the *Workflow Objects Guide*.

▶ To define the OLE automation active link action

1 In BMC Remedy Developer Studio, choose File > New > Active Link.

The New Active Link dialog box appears.

2 Select the server on which you want to create the active link, and click Finish.

The active link opens in an editor.

3 Set up conditions in the Associated Forms panel, the Execution Options panel, and the Run If Qualification panel.

4 Right-click the If Actions panel or the Else Actions panel.

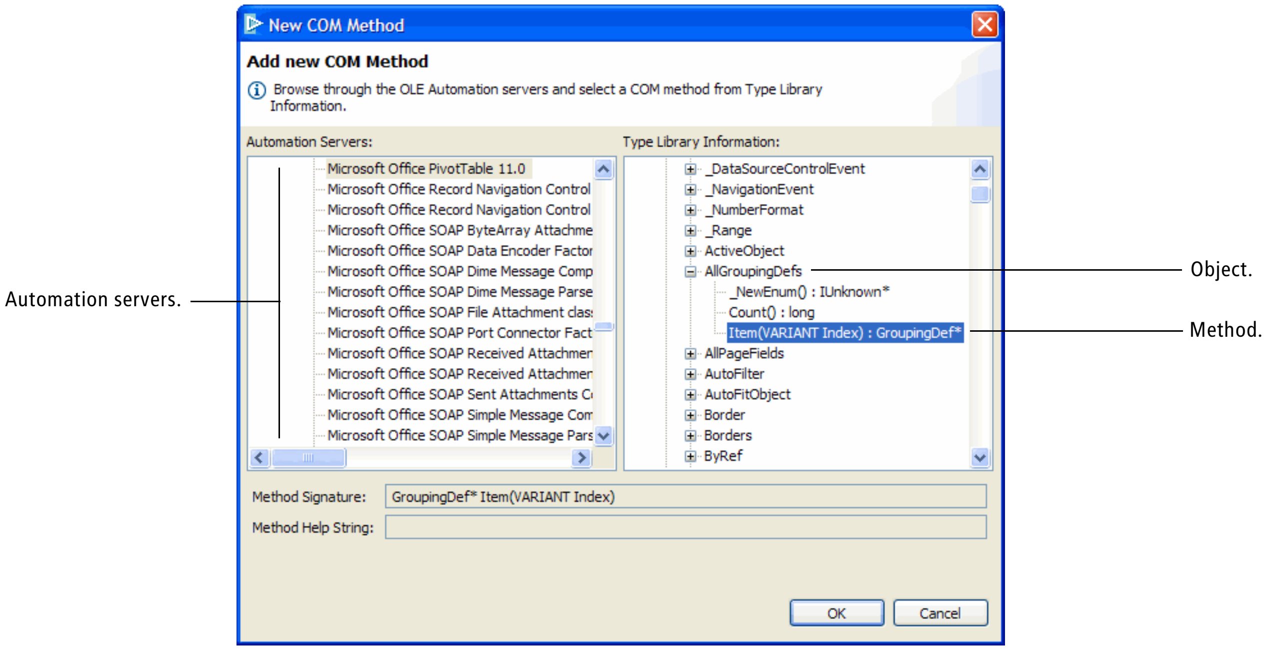
5 Choose Add Action > OLE Automation.

An OLE Automation panel appears under the If Actions or Else Actions panel.

6 In the Method tree, click the node labeled "Click here to add a new method" (Figure 20-7).

The Add New COM Method dialog box appears, as shown in Figure 20-6.

Figure 20-6: OLE Automation active link action—adding a method



Add new COM Method

Browse through the OLE Automation servers and select a COM method from Type Library Information.

Automation Servers:

- Microsoft Office PivotTable 11.0
- Microsoft Office Record Navigation Control
- Microsoft Office Record Navigation Control
- Microsoft Office SOAP ByteArray Attachme
- Microsoft Office SOAP Data Encoder Factor
- Microsoft Office SOAP Dime Message Comp
- Microsoft Office SOAP Dime Message Parse
- Microsoft Office SOAP File Attachment clas:
- Microsoft Office SOAP Port Connector Fact
- Microsoft Office SOAP Received Attachmer
- Microsoft Office SOAP Received Attachmer
- Microsoft Office SOAP Sent Attachments C
- Microsoft Office SOAP Simple Message Com
- Microsoft Office SOAP Simple Message Pars

Type Library Information:

- _DataSourceControlEvent
- _NavigationEvent
- _NumberFormat
- _Range
- ActiveObject
- AllGroupingDefs
  - _NewEnum() : IUnknown*
  - Count() : long
  - Item(VARIANT Index) : GroupingDef*
- AllPageFields
- AutoFilter
- AutoFitObject
- Border
- Borders
- ByRef

Method Signature: GroupingDef* Item(VARIANT Index)

Method Help String:

7 From the OLE Automation Servers list, select the appropriate OLE server or control:

- **OLE Local Servers**—Lists the OLE servers on your machine.
- **OLE Automation Control**—Lists the controls on your machine.

Because remote automation is not supported, users can access only the OLE active link automation action if the same OLE components selected in BMC Remedy Developer Studio also exist on their computers.

8 Double-click the appropriate server or control.

A list of relevant objects is displayed in the Type Library Information list.

9 In the Type Library Information list, expand the Objects in Library > Objects category.

A list of objects defined for the selected server or control appears. You can also expand Aliases, Enumeration, Structures, or Unions to view display-only reference information for the server or control.

10 In the Objects category, expand the appropriate object.

A list of methods for the selected object appears.

11 Select a method, and click OK to add it.

The selected method is displayed in tabular format on the OLE Automation panel. You can click the appropriate cells in the table to define method parameters and return values (see Figure 20-7).

In the Method column, each item that you can define has a placeholder such as `Type in your value here` or `Select a field here`. The placeholder is a temporary representation of the item value. If a placeholder does not appear where you would expect to find a return value or a method parameter, the item does not exist for the selected method.

If an OLE method returns an object, you can nest the methods of the object that is returned. That is, if object B is returned when you call a method on object A, you can nest these methods by using this syntax:

```
Return Value = A.B(parameter1, parameter2, ...).
```

12 Define method parameters, if applicable. Expand the Parameters category in the Method list, then find the parameter name and expand it.

The words `Type in your value here` appear below the selected parameter, as a temporary representation of the parameter value. The parameter value must be of the data type specified in the parameter name.

You can define a parameter in one of the following ways:

- Click `Type in your value here` and type the parameter value.
- Click the adjacent ellipsis button to open the Field / Keyword Selector. In the Selector, click OLE Automation Component, Fields, or Keywords. Choose the item to use for the parameter value, and click OK.
- Optional parameters are displayed in brackets ([ ]). If you leave these parameters blank, BMC Remedy User substitutes default values for them when it runs.

13 Repeat step 12 until every parameter is defined.

If the value of a parameter is another method, as described in step 11, you must define parameters for the nested method as well.

14 Define the method return value, if applicable. Expand the Method Returns > Return Value category in the Method list.

The words `Select a field here` appear as a temporary representation of the return value.

15 Click the ellipsis button adjacent to `Select a field here`, and use the Field Selector to choose a field.

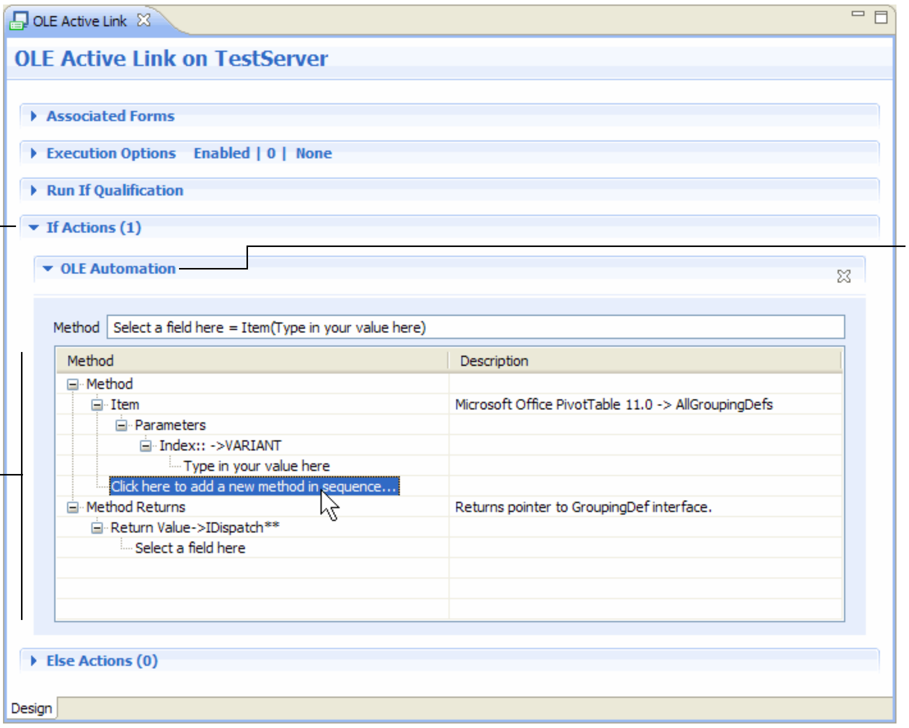
The return value must be of the data type specified in the Return Value category. If the return value is a pointer, the only way to use the returned object is to cascade or nest a method of the object being returned.

16 To delete a method, right-click the method name and choose Delete Method.

17 Add more OLE automation actions, if desired.

Figure 20-7 shows an example of an active link that is already created, and explains the elements of the OLE Automation panel.

Figure 20-7: OLE Automation active link


Active link actions.
To rearrange a list of actions, right-click one and use the Move Action controls.
As you select methods, they are displayed in a tree view. If they have parameters or return values, you can enter them in this list.
OLE Active Link
OLE Active Link on TestServer
Associated Forms
Execution Options Enabled | 0 | None
Run If Qualification
If Actions (1)
OLE Automation
Else Actions (0)
Design

Method: Select a field here = Item(Type in your value here)

Method	Description
Method	
Item	Microsoft Office PivotTable 11.0 -> AllGroupingDefs
Parameters	
Index:: ->VARIANT	
Type in your value here	
Click here to add a new method in sequence...	
Method Returns	Returns pointer to GroupingDef interface.
Return Value->IDispatch**	
Select a field here	

The Type Library Information list includes aliases, enumerations, objects, structures, and unions. OLE automation servers define these special data types, which can be used as parameters or return types for method calls defined in the objects.

Some data types are used for display only and are not useful to OLE automation authors. However, methods are defined within objects, and these methods are used to create OLE automation active link actions.

OLE automation example—Sample:SpellCheck

The `Sample:SpellCheck` active link in the Sample application uses OLE to access the spell checker of Microsoft Word. Before analyzing this active link, you must understand how objects, methods, parameters, and return values are identified in the If Action or Else Action tabs for creating OLE automation active links.

Eight OLE automation actions are used to check spelling, as follows:

- `FileNewDefault( )` creates and opens a temporary Word file.
- `AppHide( )` hides the Word application from the user.
- `Insert($Description$)` inserts the contents of the Description field into the temporary file.
- `ToolsSpelling( )` runs the spelling checker on the file.
- `EditSelectAll( )` selects the text in the document.

- `$Description$ = Selection( )` returns the checked text back to the Description field.
- `FileClose(Val(2))` closes the file.
- `AppClose( )` closes the Word application.

Finally, a Set Fields action trims the trailing carriage-return character from the end of the text. If the user is not running BMC Remedy User on a computer with Microsoft Word, an Else action displays an error message.

Issues and considerations

Keep the following issues in mind when working with OLE:

- OLE does not work on the web.
- Neither persistence nor event processing is supported by AR System. You can write your own ActiveX controls to compensate.
- Extensive troubleshooting tips for OLE Automation are available at the Customer Support website, `http://www.bmc.com/support`. When you access the knowledge base, enter the keyword OLE.
- The OLE capabilities of BMC Remedy User are designed to provide a client-side, single user integration point. For greater speed, more robust capabilities, or server-side integration, use the AR System C or Java API.
- BMC Remedy User OLE methods that return lists of forms, servers, and other objects do not work in Visual Basic Script (VBS, VBScript). This is because BMC Remedy User transfers these as string arrays, while VBScript works only with Variant arrays. This is described in Microsoft article: `http://support.microsoft.com/support/kb/articles/Q165/9/67.ASP`.
- The BMC Remedy User OLE interface is a single document interface with multiple objects.

Chapter

21 Dynamic data exchange

Dynamic data exchange (DDE) can be used for AR System client integration and, indirectly, for server integration.

The following topics are provided:

- Overview (page 286)
- Configuring your system to use DDE with AR System (page 287)
- Using active links with DDE (page 290)
- Using BMC Remedy User reporting with DDE (page 293)
- Using a DDE execute from an external application to trigger AR System (page 295)
- Using AR System with DDE, third-party applications, and macros (page 296)
- Examples (page 300)
- Issues and considerations (page 306)

Overview

Dynamic data exchange (DDE) is a Microsoft-defined method for exchanging data between Windows applications, such as Excel, Word, and Visual Basic. In DDE, data is passed between DDE clients (destinations) and DDE servers (sources) on the same machine.

The DDE client initiates the exchange by establishing a conversation with the DDE server so that it can request data or services from the server. The server responds by providing the data or services to the client. A server can have many clients at the same time, and a client can request data from multiple servers. Additionally, an application can be both a client and a server. In AR System, BMC Remedy User is usually the client application, and the DDE service is the server application. BMC Remedy User can also be a DDE server.

DDE parameters used by AR System

AR System uses the following DDE parameters to identify data exchanged during a DDE conversation:

DDE parameter	Description
`Service Name`	Identifies the DDE application with which BMC Remedy User establishes a conversation. Application DDE name.
`Topic`	Identifies the logical data context. For example, a file name or a category of command (like the `system` command in Excel).
`Item`	Specifies the location in the DDE application where you set a value. For example, the cell location in a spreadsheet. Used by the DDE `Poke` command.
`Path`	Specifies the location of DDE application executables specified in the Service Name field.
`Command`	Specifies data to send to the DDE application. The data can be a literal value or a data field value.

For more information about DDE, see the Microsoft Windows documentation.

Methods of integration

You can use DDE to integrate AR System with third-party applications in the following ways:

- Use the DDE active link action to send data to or execute a command in another Windows application. See "Using the DDE active link action" on page 291.
- Use the DDE active link keyword to request data from another Windows application. See "Using the DDE active link keyword" on page 293.

  In BMC Remedy Developer Studio, you specify a DDE command and a trigger action for an active link. Users activate the active link from a toolbar button or through actions in BMC Remedy User.
- Use DDE and BMC Remedy User to send a report to another Windows application and cause the application containing the AR System report to open. See "Using BMC Remedy User reporting with DDE" on page 293.
- Use a third-party application and DDE to launch BMC Remedy User and execute a macro. See "Using AR System with DDE, third-party applications, and macros" on page 296

For more information about DDE, see the Microsoft Windows documentation.

For more information about integrating AR System with another application, see that application's documentation.

Configuring your system to use DDE with AR System

This section outlines how to configure the DDE settings on your local machine to enable AR System to work with DDE.

Working with your dde.ini file

When a DDE client sends report data to a DDE server, the client needs a way to specify the details of the transfer, including how and where to send the data, and what to do with it when it gets there. When an AR System client is the DDE client, you use a `dde.ini` file to specify the details.

The `dde.ini` file must be in the `ARHOME` directory for the AR System client. The default location for the `ARHOME` directory on a Windows platform is `C:\home`.

Use a text editor to modify your `dde.ini` file. The `dde.ini` file uses the following syntax:

```
[SectionTag]
Path = pathToApplication
Application = DDEApplicationName
Topic = topicName
Format = formatName
XFRDATA = transferMechanism
Command1 = [DDEApplicationCommand1]
.
.
.
Commandn = [DDEApplicationCommandn]
```

The parameters in the `dde.ini` file are as follows:

Parameter	Definition
`SectionTag`	Identifies a portion of the `dde.ini` file that contains parameters pertaining to one instance of exporting a report to an application. You can have multiple sections in a file. This value must be unique in the `dde.ini` file. Use descriptive names to help you remember what each section is for. Names are not case-sensitive and cannot contain spaces. This value is used when the Report To Application button on the reporting facility window is selected to identify which section in the `dde.ini` file should be used to control the data transfer.
`Path`	Defines the full execution path name to the DDE server application. If the application is not already running, it is started by using any command-line arguments you specify as part of the Path parameter.
`Application`	Identifies the DDE server name of the DDE server application. This name is defined during the development of the application and is not configurable. For example, `excel` is the DDE server name for Excel and `winword` is the DDE server name for Word.
`Topic`	Controls how data is exchanged for some DDE server applications. For example, when you execute commands, functions, or macros for Excel, the `Topic` should be `system`.
`Format`	Defines which column separator AR System uses when multiple fields in a data record are output. This overrides any format specification in the report definition. The options are: ■ `TAB`—Values separated by a *`tab`* character. ■ `CSV`—Values separated by commas. ■ `Record`—Values output in record format. All the page setup information is reset. You can also specify an optional parameter `CharsPerLine`. If `CharsPerLine` is not specified, the default is `80`. ■ `Current`—Uses the format and page setup defined in the application.

Parameter	Definition
XFRDATA	Defines the actual data transfer mechanism. The options are: ■ Clipboard—Copy data to the Windows Clipboard and then paste it into the external application from there ■ File—Copy the data to a temporary file and then open the temporary file with the external application.
Command*n*	Defines the commands to send to the DDE server application to tell it what to do with the data transferred to it. Each DDE server application has a set of commands that it supports. For Excel, these are the Excel macro functions and Visual Basic commands. Excel commands are documented in the Excel Help system. In the dde.ini file, each specified command is enclosed in square brackets.

Sample dde.ini files

The following sample dde.ini file sends a report with values separated by tabs to Microsoft Excel. It copies report data from the clipboard and pastes it into a new document:

```
[excelclipboard]
Path=C:\excel\excel.exe
Application=excel
Topic=system
Format=Tab
XFRDATA=Clipboard
Command1=[NEW(1)] [PASTE()] [SAVE.AS(,0)]
```

BMC Remedy User creates a random file in the user's TEMP folder and substitutes this file name were %f appears. This report is then opened in Excel:

```
[excelfile]
Path=C:\excel\excel.exe
Application=excel
Topic=system
Format=Tab
XFRDATA=FILE
Command1=[OPEN("%f")]
```

The following sample dde.ini file sends a report in record format to Microsoft Word. It copies report data to the clipboard, opens a new file in Word and pastes the data into the new file:

```
[wordrecord]
Path=C:\msoffice\winword\winword.exe
Application=winword
Topic=system
Format=Record
CharsPerLine=100
XFRDATA=Clipboard
Command1=[FileNew .Template = "Normal", .NewTemplate = 0]
Command2=[Editpaste]
```

The following sample dde.ini file sends a report in the current report format to Microsoft Word. It creates a file containing report data and opens the new file using Word:

```
[WordCurrFormatFile]
Path=C:\msoffice\winword\winword.exe
Application=winword
Topic=system
Format=CurrentFormat
XFRDATA=File
Command1=[FileOpen .Name="%f"]
```

DDE time-out settings

AR System DDE operations have a time-out setting associated with them in the ar.ini file. The time-out setting indicates the amount of time that BMC Remedy User waits for a response from the third-party application. If there is no response after this set time, the DDE operation is not completed and a time-out message is displayed.

The [DDE] section and settings do not exist in your ar.ini file unless you add them. If you do not add a [DDE] section, the default values take effect. To specify a value, add the following lines to your ar.ini file:

```
[DDE]
AppResponseTimeout=N
TransactionTimeout=N
```

where *N* is the new DDE time-out setting in seconds. The parameters are as follows:

Parameter	Description
AppResponseTimeout	The maximum time in seconds to load the application into memory after starting it before AR System times out. The default setting is 30 seconds.
TransactionTimeout	The maximum time in seconds for the DDE server to respond to the requested DDE action before AR System times out. The default setting is 20 seconds.

Using active links with DDE

You can use DDE in active links as follows:

- Use the DDE active link action to send data to an external application or send a request to execute a command to an external application.
- Use the DDE active link keyword to request data from an external application.

Using the DDE active link action

You can use an active link with the DDE action to integrate AR System with an external application.

NOTE

If you design an active link for a form that is used by clients on Windows and other platforms, verify that the current platform is Windows as a condition of activating the DDE action. To do this, include a qualification that uses the `$HARDWARE$` keyword to verify the current platform. For more information about building qualifications, see the *Workflow Objects Guide*.

▶ To define the DDE active link action

1 In AR System Navigator of BMC Remedy Developer Studio, expand *serverName* > All Objects.

2 Right-click Active Links, and choose New Active Link.

3 On the Associated Forms panel, select the form or forms associated with the active link, filter, or escalation.

4 If there is more than one associated form, Identify the primary form.

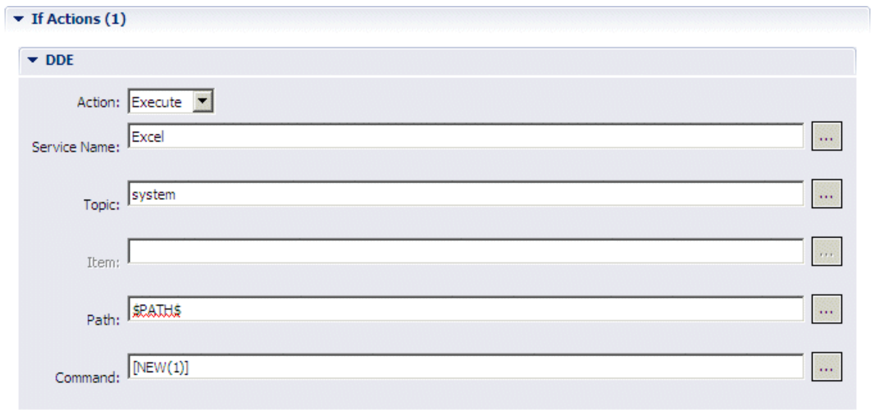
5 On the Execution Options panel and the Run If Qualification panel, set the conditions for the active link.

For more information, see the *Workflow Objects Guide*.

6 Right-click on the If Actions panel or the Else Actions panel, and select Add Action > DDE.

The fields required to define the DDE action appear. The following figure shows these fields and an example of how a DDE (Execute) active link action might look after you complete the remaining steps in this procedure.

Figure 21-1: If Actions panel with DDE action



If Actions (1)

DDE

Action: Execute

Service Name: Excel

Topic: system

Item:

Path: $PATH$

Command: [NEW(1)]

7 Complete the following fields.

Field	Description
Action	The DDE action type. The options are: ■ **Execute**—Sends a request to the DDE application to execute the commands in the Command field. The Execute action uses these parameters: ■ Service name ■ Topic ■ Path ■ Command ■ **Poke**—Use the DDE Poke action to send a request to the DDE application to set the value specified in the Command field to the location (for example, a field or cell) specified in the Item field. The Poke action uses these parameters: ■ Service name ■ Topic ■ Item ■ Path ■ Command See "DDE parameters used by AR System" on page 286.
Service	Unique ID of the DDE application, for example, `excel` or `winword`. See "`Service Name`" on page 286.
Topic	DDE topic. See "`Topic`" on page 286.
Item	DDE item (if applicable). The Item field is enabled if the Action type is Poke.
Path	Location of the service. See "`Path`" on page 286.
Command	Command to execute. See "`Command`" on page 286. You must enter a value in this field. If the action is Execute, a command is sent to a DDE application. If the action is Poke, data is set in the DDE application. See the *Workflow Objects Guide* for information about loading the result of a DDE request operation into a field by using an active link that performs a Set Fields action.

AR System Active Link Support for DDE Execute and Poke

The DDE action supports the following functions:

- **DDE Execute**—Causes the DDE server application to execute a process or command. For example, a DDE Execute function might direct Excel to start up and run a particular Excel macro.
- **DDE Poke**—Sends a piece of data from the DDE client to the DDE server. For example, a DDE Poke function might take a value from a field on an AR System form and poke (that is, write or copy) it into a cell in an Excel spreadsheet or a bookmark in a Word document.

Using the DDE active link keyword

Use the Set Fields active link action to set the value of a field. Use the `$DDE$` keyword with a Set Fields action to cause a DDE Request to be made to another application to obtain the data to fill in the field. For example, a workflow definition could look up the current price of an item in a price list maintained as an Excel spreadsheet and copy the value into a field on a form.

When the active link is performed on a Windows client, the specified DDE request is executed and BMC Remedy User waits for the operation to complete. The data that the DDE request returns is then entered into the field.

If the active link is triggered by an action in a form on a non-Windows client, an empty or null string is returned.

The `$DDE$` keyword indicates that all following text is a DDE command line. The command line can include substitution parameters from the current screen to enable values from the current screen to be placed into the command line before it is executed. You can select substitution parameters (and the `$DDE$` string) from the Fields Value list.

The syntax of the `$DDE$` keyword is as follows:

```
$DDE$ serviceName;topic;path[;item]
```

For example, the following operation returns the contents of cell R1C1 of a file named `sheet1` in Microsoft Excel to the current field:

```
$DDE$ excel;sheet1;C:\excel\excel.exe;R1C1
```

For more information about the parameters you can use for the `$DDE$` keyword, see "DDE parameters used by AR System" on page 286.

Using BMC Remedy User reporting with DDE

For exporting large blocks of data out of AR System and into another Windows application using DDE, BMC Remedy User reports are the most effective mechanism.

Before you send an AR System report to another Windows application, perform the following tasks:

Step 1 If necessary, create or modify a `dde.ini` file. See "Working with your dde.ini file" on page 287.

Step 2 Configure BMC Remedy User to allow report data to be passed to an external application.

Step 3 Create a report that gathers the desired data from the AR System database.

Configuring BMC Remedy User to pass report data

By default, BMC Remedy User is not configured to pass data to an external application using DDE. To enable this function, you must modify a BMC Remedy User option.

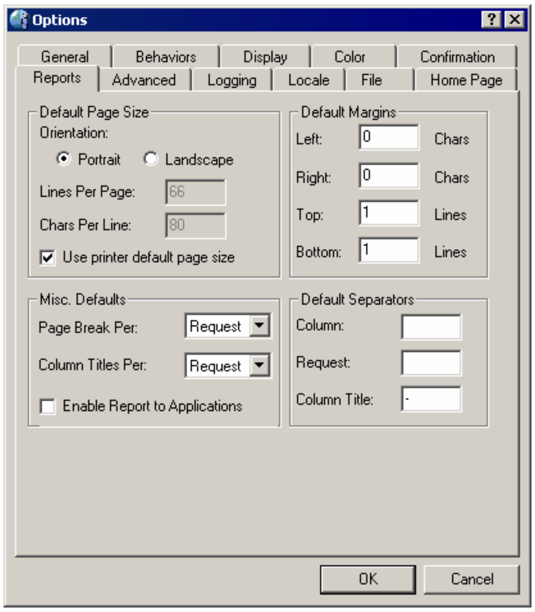
▶ To configure the BMC Remedy User to pass report data

1 Under the Tools menu, select Options.

2 Click the Reports tab.

3 Select the Enable Report to Application check box, and click OK.

Figure 21-2: BMC Remedy User Options window for reporting



Default Page Size
Orientation: [✓ Portrait] Landscape
Lines Per Page: 66
Chars Per Line: 80
[✓] Use printer default page size

Default Margins
Left: 0 Chars
Right: 0 Chars
Top: 1 Lines
Bottom: 1 Lines

Misc. Defaults
Page Break Per: Request
Column Titles Per: Request
[ ] Enable Report to Applications

Default Separators
Column:
Request:
Column Title: -

Creating a report for DDE export

Create an AR System report to export large amounts of data using DDE.

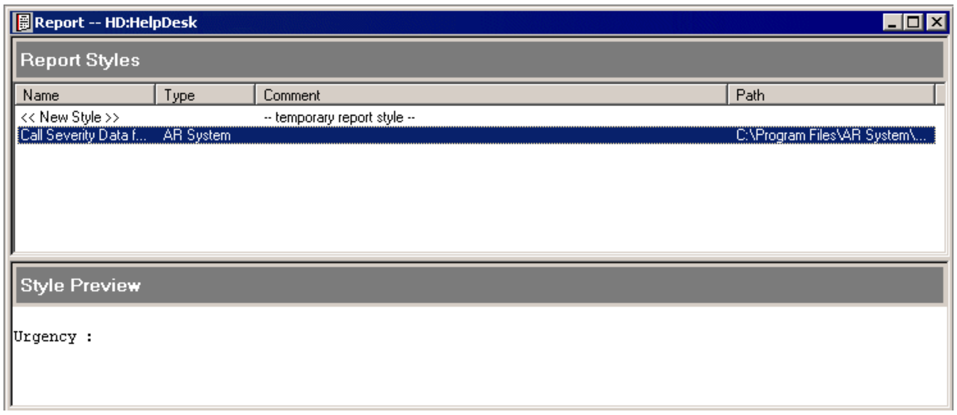
▶ To create a report for DDE export

1 Open the form that contains the data to export.

2 Choose Tools > Reporting.

The Report window appears.

Figure 21-3: BMC Remedy User Report window



Report -- HD:HelpDesk

Report Styles

Name	Type	Comment	Path
<< New Style >>		-- temporary report style --	
Call Severity Data f...	AR System		C:\Program Files\AR System\...

Style Preview

```
Urgency :
```

3 Create a report, or select an existing report from the list.

Make sure the report you create contains the data to export to DDE.

4 Choose Report > Export to > Application.

The Report to Application dialog box appears

5 Select an entry from the Application name menu.

The entries in the Application name menu correspond to the sections in your `dde.ini` file. For more information about the dde.ini file, see “Working with your dde.ini file” on page 287.

The report data is passed to the DDE application according to the control parameters in a `dde.ini` file.

The data from the report is processed in the DDE application.

Using a DDE execute from an external application to trigger AR System

Another Windows application can trigger BMC Remedy User as a DDE server. BMC Remedy User supports only one DDE command, `RunMacro`, which causes BMC Remedy User to execute a named macro and accept any passed parameters as input to the macro.

- Application (Service) Name—ARUSER-SERVER
- Topic Name—DoExecMacro
- Item Name—(none)
- Command Name—RunMacro

The following code sample shows the exact syntax of the `runmacro` command that the DDE client issues to AR System.

```
[RunMacro(macroPath,macroName,parameterName1=parameterValue1,
    parameterName2=parameterValue2,...)]
```

Parameter	Description
macroPath	The fully qualified path to the directory on the PC where AR System macros are stored (for example, `C:\home\arcmds`).
macroName	The complete name of the macro as defined in AR System (not the name of the file in the `macroPath` directory).
parameterName	The name of a parameter that was recorded into the macro.
parameterValue	The value to substitute.

Note the following guidelines when using a `RunMacro` command:

- None of the parameters should contain a comma.
- There are no spaces in the command statement.
- The square brackets around the statement are required.

Using AR System with DDE, third-party applications, and macros

Third-party applications can use a DDE program to send a request to execute a macro in BMC Remedy User. This program must include the following components:

- DDE server name of BMC Remedy User
- Path for BMC Remedy User
- DDE topic BMC Remedy User supports
- DDE function BMC Remedy User supports

DDE server name and BMC Remedy User path

The DDE server name and the path of BMC Remedy User are added to your `win.ini` file when you install BMC Remedy User. This information is added to the `[AR System]` section, as follows:

```
AppName=ARUSER-SERVER
ProgramPath=pathToaruser\aruser.exe DDEcall pathToaruser
```

The meanings of the fields are as follows:

- `AppName`—The DDE server name for BMC Remedy User.
- `ProgramPath`—The path for BMC Remedy User.

The path is needed so that a third-party application can find and start BMC Remedy User. Use this field exactly as shown.

Also, you must complete one of the following tasks in your DDE program before executing BMC Remedy User:

- Set your `PATH` environment variable to the BMC Remedy User directory.
- Change to the BMC Remedy User directory.

Supported DDE topic and function

BMC Remedy User supports the `DoExecMacro` DDE topic, which enables you to use DDE to run macros in AR System through third-party applications.

BMC Remedy User also supports the `RunMacro` function, which creates a buffer that contains the path and name of the macro along with any needed parameters. This buffer contains the information that BMC Remedy User needs to find and run the macro.

Example program and buffer

The following example C program sends a DDE message to BMC Remedy User, instructing it to run a macro called `SendMessage` found in the `C:\app\macro` directory. This macro requires two parameters:

- The name of the user that receives the message
- The message text

The `RunMacro` function in this example creates the following buffer:

```
[RunMacro(C:\app\macro,SendMessage,
Name=John Smith,Contents=Don't forget our meeting on Friday)]

/* DoDDEInit -- This routine initializes dde conversation. It must
be called before any dde conversation can happen.
*/
BOOL WINAPI DoDDEInit(void)
{
BOOL bResult = FALSE;
// Read the path to the aruser.exe
if (GetProfileString("ARSYSTEM",
"ProgramPath","",szARuserPath,
sizeof(szARuserPath) - 1) == 0 ) {
// display an error message if aruser is not installed.
return FALSE;
}
// Initialize the dde client
if (lpDdeProc = MakeProcInstance((FARPROC)DdeCallBack, hInst)) {
idInst = 0;
if (DdeInitialize((LPDWORD)&idInst, (PFNCALLBACK)
lpDdeProc, DDE_INIT_FLAGS, NULL) == DMLERR_NO_ERROR){
Hszize();
bResult = TRUE;
}
else
FreeProcInstance((FARPROC)lpDdeProc);
}
return (bResult);
} // DoDDEInit()
/* DoDDEUnInit -- Uninitializes applications and frees call back
*/
VOID WINAPI DoDDEUnInit(void)
{
if (hConv) {
DdeDisconnect(hConv);
hConv = NULL;
}
if (lpDdeProc) {
DdeUninitialize(idInst);
FreeProcInstance((FARPROC)lpDdeProc);
}

UnHszize();

} // DoDDEUnInit()

/* Hszize -- This creates often used global hszs from standard
global strings.
It also fills the hsz fields of the topic and item tables.
*/
static void Hszize(void)
{
char szServerName[MAX_TOPIC + 1];

// Get the name of server in string handle format
GetProfileString("ARSYSTEM","AppName","",
szServerName,MAX_TOPIC);
hszServerName = DdeCreateStringHandle(idInst,
szServerName,0);

// For the get details topic, get its string handle format
hszExecMacroTopic = DdeCreateStringHandle(idInst,
"DoExecMacro", NULL);

} // Hszize()

/* UnHszize -- This destroys often used global hszs from standard
global strings.
*/
static void UnHszize(void)
{
DdeFreeStringHandle(idInst, hszServerName);

DdeFreeStringHandle(idInst, hszExecMacroTopic);

} // UnHszize()
```

```
/* DoDDERunMacro -- This routine starts a dde conversation with
aruser and sends its run macro command.
*/
VOID WINAPI DoDDERunMacro()
{
hConv = 0;

// Start the connection for run macro with the aruser server.
// NOTE: when we use NULL for the pCC parameter DDEMEL sends
// the default CONVECONTEXT.
while (TRUE) {
hConv = DdeConnect(idInst,hszServerName,
hszExecMacroTopic,NULL);
if (hConv)
break;                          // a connection was established
if (DdeGetLastError(idInst) != DMLERR_NO_CONV_ESTABLISHED)
break;
// Try again, maybe by now aruser is up and running.
} //while (TRUE)

if (hConv) {
// Build the a buffer that contains RunMacro function and
// send it to the aruser server
char szExecute[255];
HDDEDATA hddeExecute;

// construct the data to be passed to the data
wsprintf(szExecute,"[RunMacro(%s,%s,%s=%s,%s=%s)]",
(LPSTR)"C:\\app\\macro",    // the path where the macro is
(LPSTR)"SendMessage",       // This is the name of the macro
// to run
(LPSTR)"Name",              // This is the parameter name
(LPSTR)"John Smith",        // This is the parameter value
(LPSTR)"Content",           // Parameter name
(LPSTR)"Don't forget about our meeting on Friday");

if (!(hddeExecute = DdeCreateDataHandle(idInst,
(LPVOID)szExecute,
lstrlen(szExecute)+1,0,NULL,CF_TEXT,NULL)))
;  // give a memory allocation error message
else {
DdeClientTransaction((LPBYTE)hddeExecute, -1, hConv,
NULL,CF_TEXT, XTYP_EXECUTE, TIMEOUT_ASYNC, &XactID);
DdeFreeDataHandle(hddeExecute);
}
}//if (hConv)
else {
// failed to connect
}
} // end DoDDERunMacro()
```

The following examples show macro programs in Excel, Word, and Visual Basic that run the SendMessage macro in BMC Remedy User. The macros send a message to John Smith, reminding him of a meeting on Friday. These sample programs assume that BMC Remedy User is running.

Excel macro

```
Sub RunMacro()
Dim RunMacroString As String
RunMacroString = "[RunMacro(C:\app\macro,Send Message,Name=John
Smith,Contents=Don’t forget our meeting on Friday)]"
channelNumber = DDEInitiate("ARUSER-SERVER", "DoExecMacro")
DDEExecute channelNumber, RunMacroString
DDETerminate channelNumber
End Sub
```

Word macro

```
Sub MAIN
RunMacroString$ = "[RunMacro(C:\app\macro,Send Message,Name=John
Smith,Contents=Don’t forget our meeting on Friday)]"
channelNumber = DDEInitiate("ARUSER-SERVER", "DoExecMacro")
DDEExecute channelNumber, RunMacroString$
DDETerminate channelNumber
End Sub
```

Visual Basic macro

```
Private Sub cmdExecute_Click()
Dim RunMacroString As String
' Application|Topic
txtMacroPath.LinkTopic = "ARUSER-SERVER|DoExecMacro"
txtMacroPath.LinkMode = 2
RunMacroString = "[RunMacro("C:\app\macro,SendMessage,Name=John
Smith,Contents=Don’t forget our meeting on Friday)]"
txtMacroPath.LinkExecute RunMacroString 'send DDE message
End Sub
```

Examples

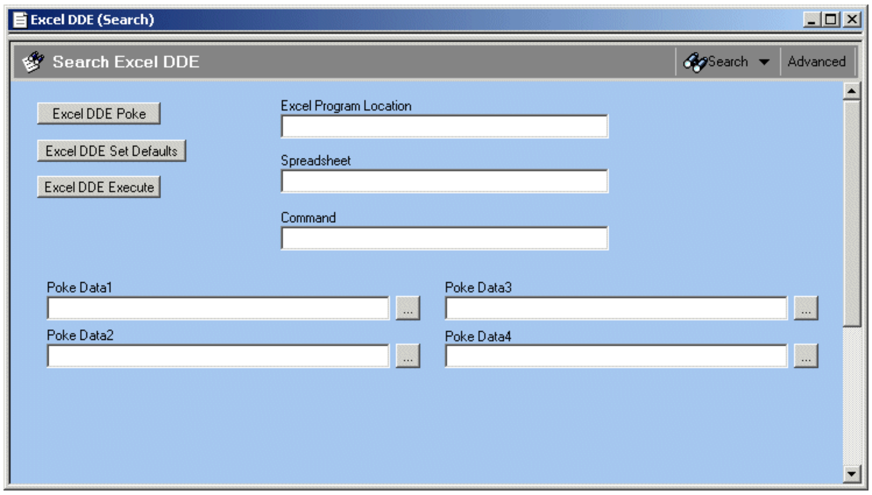
This section contains examples of integrating AR System with DDE applications.

Integrating with Microsoft Excel

The following example includes two active links that use DDE to integrate Microsoft Excel with AR System. Using these active links, a user can open and populate a spreadsheet in Microsoft Excel through an AR System form.

The following figure shows a form with active links that work with Microsoft Excel. The three buttons on the left of the form activate the active links that open and populate a spreadsheet. The active links reference the fields to determine where Microsoft Excel is installed, which spreadsheet to open, and what data to send to the spreadsheet.

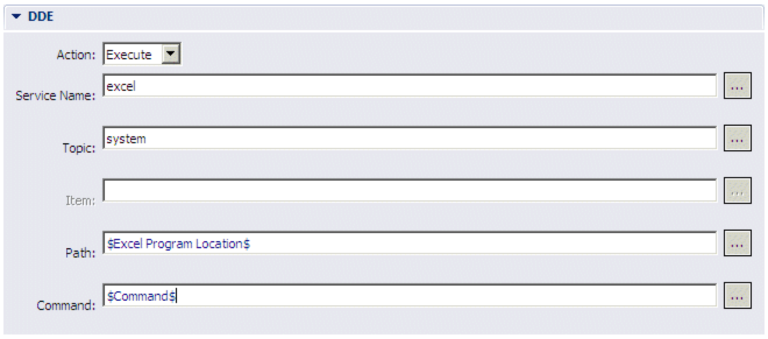
Figure 21-4: Sample DDE form



When the user clicks Excel DDE Execute, the first active link is executed and the specified spreadsheet is opened in Microsoft Excel.

When creating this active link, define a DDE Execute If Action as shown in Figure 21-5.

Figure 21-5: Sample active link—excel DDE execute



Action:	Execute
Service Name:	excel
Topic:	system
Item:	
Path:	$Excel Program Location$
Command:	$Command$

- The Action field instructs the active link to execute a command.
- The Path field references the Excel Program Location field in the sample form to determine where Microsoft Excel is installed.
- The Command field references the Command field in the sample form to determine the specific action to perform.

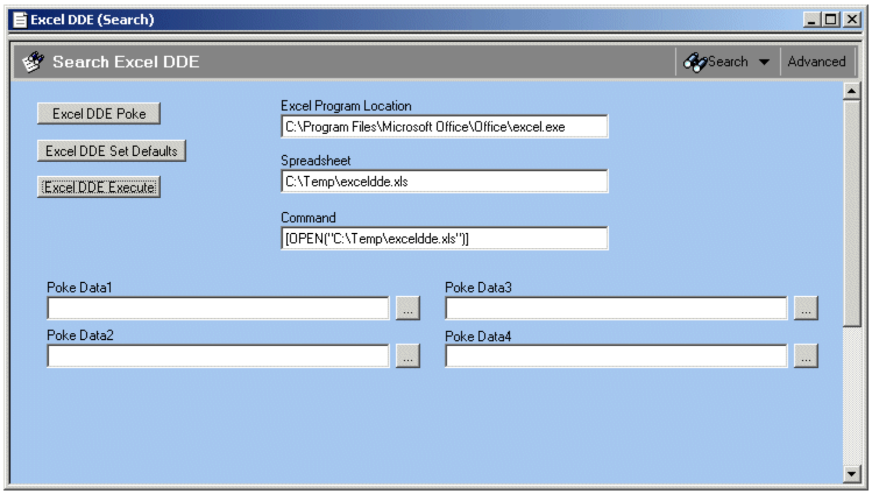
The Command field in the sample form must have a value for this active link to activate. The qualification is as follows:

```
'Command' != $NULL$
```

In the sample form, a user enters the location of the Microsoft Excel program in the Excel Program Location field, a specific spreadsheet in the Spreadsheet field, and an `OPEN` command in the Command field. You might want to enable your users to populate these fields through active links or menu lists to make sure that the syntax is correct.

As shown in Figure 21-6, a user's Microsoft Excel executable is located at `C:\Program Files\Microsoft Office\Office\excel.exe`. The particular spreadsheet to open is located at `C:\Temp\exceldde.xls`. The `OPEN` command in the Command field opens the spreadsheet.

Figure 21-6: Sample DDE form with sample data



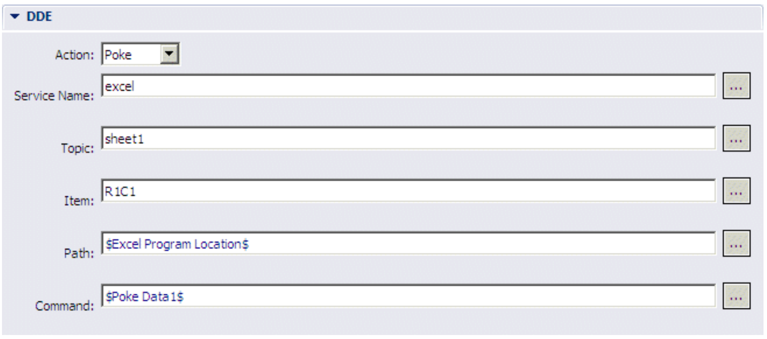
Excel Program Location: C:\Program Files\Microsoft Office\Office\excel.exe
Spreadsheet: C:\Temp\exceldde.xls
Command: [OPEN("C:\Temp\exceldde.xls")]
Poke Data1:
Poke Data2:
Poke Data3:
Poke Data4:

When the user clicks Excel DDE Execute, Microsoft Excel opens and displays the `exceldde.xls` spreadsheet.

When the user clicks Excel DDE Poke, the second active link is executed and a specified spreadsheet is populated.

When creating this active link, define a DDE Poke action, as shown in Figure 21-7.

Figure 21-7: Sample active link—excel DDE poke



Action: Poke
Service Name: excel
Topic: sheet1
Item: R1C1
Path: $Excel Program Location$
Command: $Poke Data1$

- The Action field instructs the active link to send data to a specific location.
- The Item field indicates the item to which data is poked.
- The Path field references the Excel Program Location field in the sample form to determine where Microsoft Excel is installed.
- The Command field references the Poke Data1 field in the sample form to determine the data to send to cell R1C1 in the spreadsheet.

The Excel Program Location, Spreadsheet, and Poke Data 1 fields in the sample form must all have values for this active link to activate. The qualification is as follows:

```
(('Excel Program Location' != $NULL$ ) AND
('Spreadsheet' != $NULL$ )) AND ('Poke Data1' != $NULL$ )
```

There are three actions for this sample active link—one each to poke data from the three poke fields to three cells in a spreadsheet.

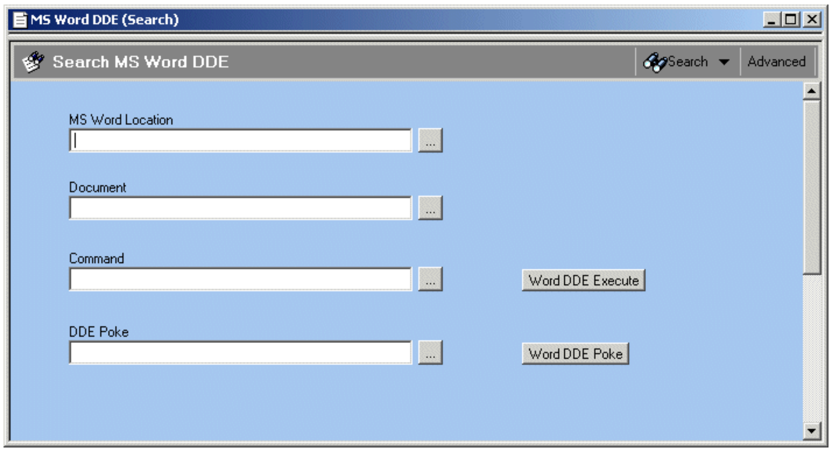
When a user supplies data to the three Poke Data fields in the sample form and clicks Excel DDE Poke, the data in the Poke Data fields are sent to cells in the spreadsheet specified in the Spreadsheet field.

Integrating with Microsoft Word

The following example includes two active links that use DDE to integrate Microsoft Word with AR System. Using these active links, a user can open and write to a document in Microsoft Word through an AR System form.

The following figure shows a sample form having active links that work with Microsoft Word. The two buttons on the right of the form execute the active links that open Microsoft Word and write to a document. The active links reference the fields to determine where Microsoft Word is installed, which document to open, and what data to send to that document.

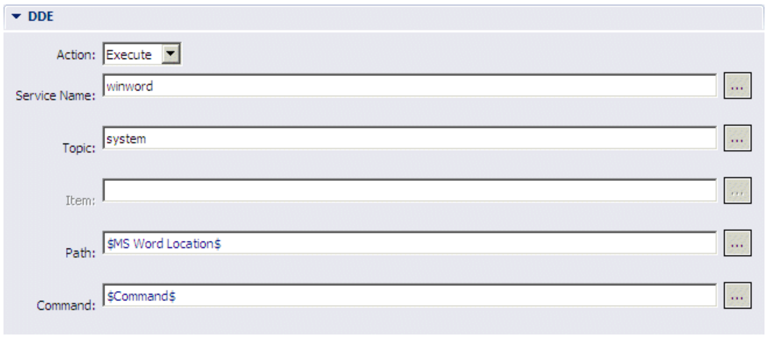
Figure 21-8: Sample DDE form



When the user clicks Word DDE Execute, the first active link is executed, and a specified document is opened in Microsoft Word.

When creating active link, define a DDE Execute action, as shown in Figure 21-9.

Figure 21-9: Sample active link—Word DDE execute



Action: Execute
Service Name: winword
Topic: system
Item:
Path: $MS Word Location$
Command: $Command$

- The Action field instructs the active link to execute a command.
- The Path field references the Word Program Location field in the sample form to determine where Microsoft Word is installed.
- The Command field references the Command field in the sample form to determine the specific action to perform.

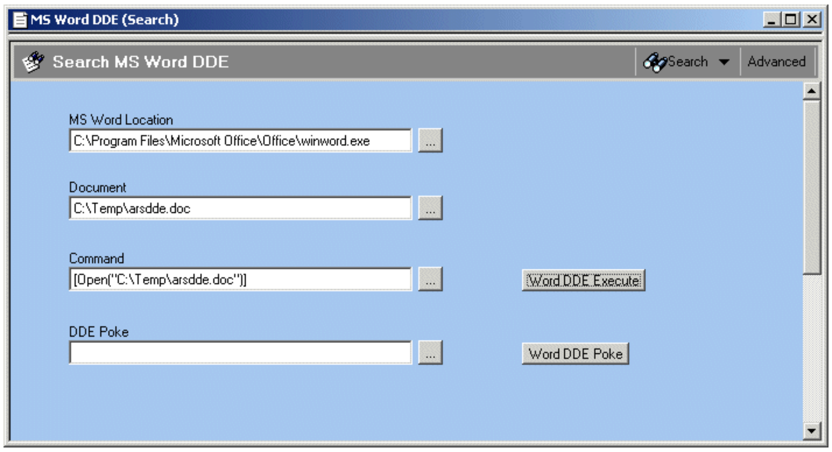
The Command field in the sample form must have a value for this active link to activate. The qualification is as follows:

```
'Command' != $NULL$
```

In the sample form, a user enters the location of the Microsoft Word program in the Word Program Location field, a specific document in the Document field, and an `OPEN` command in the Command field. You might want to enable your users to populate these fields through active links or menu lists to make sure that the syntax is correct.

As shown in the following figure, a user's Microsoft Word executable is located at `C:\Program Files\Microsoft Office\Office\winword.exe`. The particular document to open is located at `C:\Temp\arsdde.doc`. This document is opened by the `OPEN` command in the Command field.

Figure 21-10: Sample DDE form with sample data



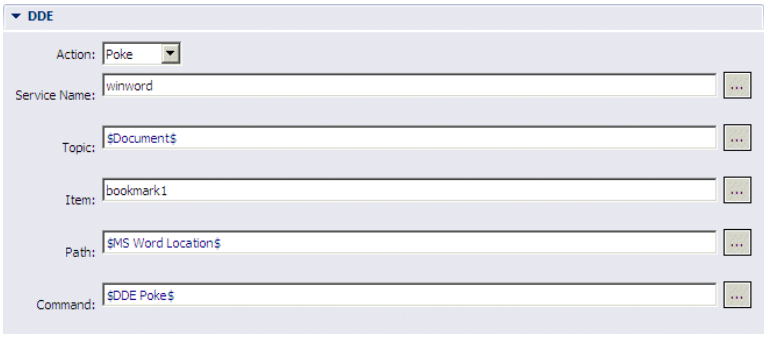
MS Word Location: C:\Program Files\Microsoft Office\Office\winword.exe
Document: C:\Temp\arsdde.doc
Command: [Open("C:\Temp\arsdde.doc")]
DDE Poke:

When the user clicks Word DDE Execute, Microsoft Word opens and displays the `arsdde.doc` document.

When the user clicks Word DDE Poke, the second active link is executed and writes to a specified document.

When creating this active link, define a DDE Poke action, as shown in the following figure.

Figure 21-11: Sample active link—Word DDE poke



Action:	Poke
Service Name:	winword
Topic:	$Document$
Item:	bookmark1
Path:	$MS Word Location$
Command:	$DDE Poke$

- The Action field instructs the active link to send data to a specific location.
- The Item field indicates the item to which data is poked.
- The Path field references the Word Program Location field in the sample form to determine where Microsoft Word is installed.
- The Command field references the DDE Poke field in the sample form to determine the data to send to bookmark1 in the document.

The Word Program Location, Document, and DDE Poke fields in the sample form must all have values for this active link to activate. The qualification is as follows:

```
(('Word Program Location' != $NULL$ ) AND
('Document' != $NULL$ )) AND ('DDE Poke' != $NULL$ )
```

When a user supplies data to the DDE Poke field in the sample form and clicks Word DDE Poke, the data in the DDE Poke field is sent to the document specified in the Document field.

Using DDE to pass data to Excel for graphing

Assume that you have an Excel spreadsheet with a recorded macro called `ChartARSystemData2Col` that generates a multiline bar chart. To use DDE to pass data to Excel:

Step 1 Create a `dde.ini` file that controls the transfer of data from AR System to Excel as shown in the following code sample.

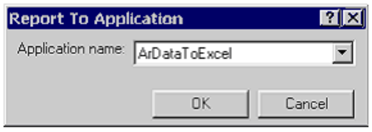
Step 2 Define an AR System report that extracts the data to graph with Excel.

Example dde.ini

```
[ArDataToExcel]
;This section uses a temp file for temporary storage
Path = C:\program~1\micros~1\office\excel.exe
Application = excel
Topic = system
Format = TAB
XFRDATA = File
Command1 = [OPEN("%f")]
Command2 = [RUN("PERSONAL.XLS!ChartARSystemData2Col")]
```

In BMC Remedy User, choosing Report > Export to > Application displays the Report To Application dialog box. The application name is the section tag from the `dde.ini` file.

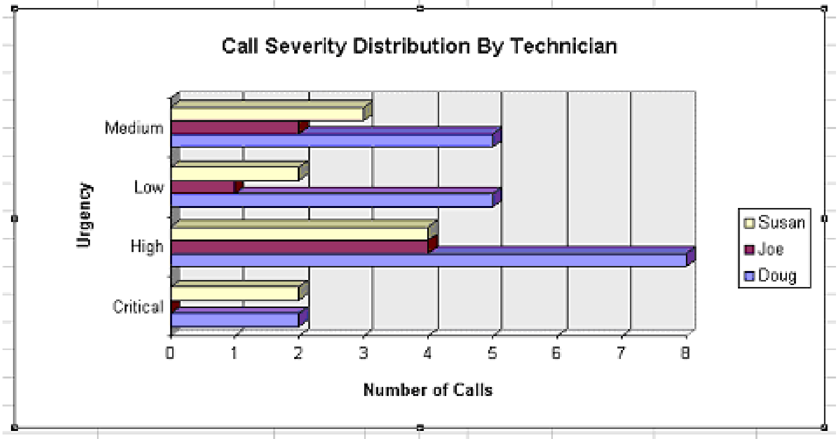
Figure 21-12: Report To Application dialog box



Application name: ArDataToExcel

The data is sent to Excel, an Excel macro is run (`PERSONAL.XLS!ChartARSystemData2Col`), and a graph is generated, as shown in Figure 21-13.

Figure 21-13: Call distribution by support technician



Issues and considerations

- DDE is supported on Windows platforms only. It is not supported on the web.
- DDE is being replaced by OLE. DDE is no longer supported by Microsoft and is provided for compatibility only.

Chapter

22 Simple network management protocol

You can use the Simple Network Management Protocol (SNMP) to monitor AR System using BMC Remedy SNMP Agent. This section describes how to configure BMC Remedy SNMP Agent if you did not configure it during installation.

The following topics are provided:

- Overview (page 308)
- BMC Remedy SNMP Agent functions (page 309)
- Sending traps (page 310)
- SNMP configuration (page 311)
- The arsnmpd configuration file (page 312)
- The snmpd configuration file (page 317)
- The armonitor configuration file (page 318)
- Starting BMC Remedy SNMP Agent (page 318)
- Stopping BMC Remedy SNMP Agent (page 319)
- Troubleshooting (page 319)

Overview

Simple Network Management Protocol (SNMP) is a protocol that network administrators use to manage complex networks through SNMP-compliant management consoles to monitor network devices.

During the AR System installation, you are prompted to configure BMC Remedy SNMP Agent. You must configure BMC Remedy SNMP Agent before you can run it. If you did not configure SNMP during installation or want to change your existing configuration, use the instructions in this guide or visit `http://www.net-snmp.org`. Check with your network administrator regarding what specific configuration settings to use.

Network administrators and AR System administrators can use BMC Remedy SNMP Agent to monitor AR System and change its state.

The BMC Remedy SNMP Agent supports the following versions of SNMP:

- Version 1 (community-based)
- Version 2c (community-based)
- Version 3 (user-based)

It supports the following levels of user-based authentication:

- No authentication, no privacy (noAuthNoPriv)
- Authentication only, no privacy (authNoPriv)
- Authentication with privacy (authPriv)

BMC Remedy SNMP Agent was developed using the net-snmp software toolkit, version 5.0.7. (For more information about the net-snmp toolkit, see `http://www.net-snmp.org/`.) The agent runs as a separate process on the same system as the AR System server, and supports the following basic SNMP operations:

- `get`
- `set`
- `get-next`
- `get-bulk` (supported in SNMP v2c and v3)
- `trap`
- `notification` (SNMP v2c, SNMP v3)

BMC Remedy SNMP Agent is compatible with all platforms that Remedy supports in this release. For more information about product compatibility, see the product compatibility matrix: `http://www.bmc.com/support`.

BMC Remedy SNMP Agent functions

You can use BMC Remedy SNMP Agent to monitor AR System, to check the state of AR System, and to send traps (notifications) when the status of any AR System process changes.

Monitoring AR System

BMC Remedy SNMP Agent can be configured to monitor AR System server statistics, AR System state, and select MIB-II data.

AR System server statistics

The statistical operations that the agent monitors are the same statistics that are available in the Server Statistics form. For more information about these statistics, see the *Optimizing and Troubleshooting Guide*.

AR System state

BMC Remedy SNMP Agent monitors the state of AR System (up or down), through the use of the managed object `arsState` (1.3.6.1.4.1.10163.1.2.1.3.0). The current value of the managed object `arsState` is used to indicate the current state of AR System. When queried, a value of 1 indicates that AR System is running; a value of 2 indicates that AR System is down.

The managed object `arsState` is also writable, so the value of `arsState` can be changed by an SNMP set operation (provided the proper user name or community string is supplied). Changing the value of `arsState` from 1 to 2 instructs BMC Remedy SNMP Agent to stop AR System. Changing the value of `arsState` from 2 to 1 instructs the agent to start AR System.

BMC Remedy SNMP Agent can monitor the following AR System processes:

- AR System server
- AR System plug-in
- BMC Remedy Email Engine
- BMC Remedy Distributed Server Option (DSO)
- AR Monitor

If any of these process changes its state (for example, if a process becomes inactive), the agent sends a trap (or a notification) to a trap receiver.

MIB-II

AR System supports the following objects in MIB-II:

- System data (for example, system description and system location)
- SNMP data and statistics

To query other objects, such as IP traffic or TCP traffic, use the SNMP agent included with your operating system. Managed objects in these sections of MIB-II are not supported by BMC Remedy SNMP Agent.

Remedy MIB

The Remedy MIB file (`Remedy-ARS-MIB.txt`) defines all the objects managed by BMC Remedy SNMP Agent and is necessary for querying Remedy specific objects by name from your SNMP client.

The `Remedy-ARS-MIB.txt` file currently defines only AR System controls, statistics, and traps. However, as it is designed for extensibility, other branches in the `Remedy-ARS-MIB.txt` file are reserved for future use.

Sending traps

A *trap* is an asynchronous message that BMC Remedy SNMP Agent sends to clients when specific events occur. The agent can be configured to send traps to a trap receiver (such as a network management station) when the state of AR System, specifically the `armonitor` process (or any AR System process, such as AR System server, AR System plug-in server, DSO, or email engine) changes. You can add a list of trap receivers (clients that receive traps) to the `arsnmpd.cfg` file.

BMC Remedy SNMP Agents supports the following trap types:

- `coldstart`—Sent when the agent starts.
- `authentication failure`—Sent when a bad community string is supplied with an SNMP request. This type of trap is supported only by SNMP versions 1 and 2c and must be enabled in the `arsnmpd.cfg` file.
- `arsStateChange`—A Remedy enterprise-specific trap type. Sent when a change of state occurs for any of these AR System processes: AR monitor, AR System server, AR System plug-in server, BMC Remedy Email Engine, and DSO.

Each trap contains the following information:

- The name of the process that changes state (for example, AR System plug-in server)
- The name of the AR System server associated with the process
- The state of the process (active =1, inactive =2)

  When a monitored AR System process changes state from *running* to *down*, the trap contains a value of `2` for `arsState`. When the process resumes, the trap contains a value of `1` for `arsState`.

NOTE

BMC Remedy SNMP Agent continues to run even if the processes it monitors are not running.

For more information about configuring traps in the `arsnmpd` configuration file, see “Trap configuration” on page 316.

SNMP configuration

NOTE

For information about configuring BMC Remedy SNMP Agent during installation, see the *Installation Guide.*

BMC Remedy SNMP Agent was built using the Net-SNMP toolkit (version 5.0.7). This section describes a subset of the more user-friendly and commonly used configuration options provided by the Net-SNMP toolkit (version 5.0.7). For information about additional configuration directives and options, see the Net-SNMP website at `http://www.net-snmp.org`.

BMC Remedy SNMP Agent uses the following configuration files:

Configuration file	Location	Purpose
Windows: `arsnmpd.cfg` **UNIX:** `arsnmpd.conf`	**Windows:** *`ARSystemServerInstallDir`*`\conf` **UNIX:** `/usr/ar/`*`ARSystemName`*`/conf`	Stores system information, access control information, and trap settings.
Windows: `snmpd.conf` **UNIX:** `snmpd.conf`	**Windows:** *`ARSystemServerInstallDir`*`\conf` **UNIX:** `/usr/ar/`*`ARSystemName`*`/conf`	Stores engine ID, number of BMC Remedy SNMP Agent reboots, and SNMP v3 user account information.
Windows: `armonitor.cfg` **UNIX:** `armonitor.conf`	**Windows:** *`ARSystemServerInstallDir`*`\conf` **UNIX:** `/etc/arsystem/`*`serverName`*	Enables BMC Remedy SNMP Agent to monitor AR System and to be started by `armonitor`.

BMC Remedy SNMP Agent uses the information in the `arsnmpd` and `snmpd` configuration files to initialize when the agent starts; therefore, you must restart BMC Remedy SNMP Agent after you make changes to the configuration files. In addition, you must restart AR System if you make changes to the `armonitor` configuration file.

NOTE

If you perform an SNMP set operation to change the value of `versionUpdateConfig.0 (.1.3.6.1.4.1.2021.100.11.0)` to 1, BMC Remedy SNMP Agent rereads the `arsnmpd.cfg` (`arsnmpd.conf`) file, so in this case you do not need to restart BMC Remedy SNMP Agent.

The arsnmpd configuration file

Use the `arsnmpd.conf` (`arsnmpd.cfg`) file to configure any of the following information:

- System information (page 313)
- Access control information, which includes community strings and users (page 313)
- Trap configuration, which identifies the systems to which trap messages are sent (page 316)
- Location of the `armonitor` configuration file (page 316)

To configure any of these items, apply a configuration directive and related arguments. You can also add comments to any configuration file by beginning any comment line with a hash [#] character. The standard syntax is as follows:

```
directive argument [optionalArgument]
# This is a comment
```

The following conditions apply to directives:

- Each directive must occupy its own line in the configuration file.
- Directives can be included in any order.
- Only one instance of a directive is permitted in a configuration file unless otherwise indicated.
- Directives are considered optional unless otherwise specified.

If you configured SNMP during the installation process, many of the configuration options are represented in this `arsnmpd` configuration file. If you did not configure SNMP during installation, a sample `arsnmpd.conf` file with comment lines and sample directives is installed. In this case, you need to remove the hash (#) characters, and provide valid arguments to the various directives. You can also add configuration directives where appropriate.

System information

The following system information can be defined in the `arsnmpd` configuration file:

Directive	Description
`syslocation`	A string representing the location of the system running BMC Remedy SNMP Agent.
`syscontact`	A string representing contact information for AR System, BMC Remedy SNMP Agent, or both.

To define the system location, add the following directive:

```
syslocation systemLocation
```

To define the system contact, add the following directive:

```
syscontact systemContactInformation
```

The argument to `syslocation` or `syscontact` can include spaces. However, all the information must be on the same line and no longer than 255 characters.

For example:

```
syslocation Lab in room 101
syscontact Call Joe at 555-5555 or joe@mail.com
```

You can access information defined by these directives from BMC Remedy SNMP Agent by querying the related MIB-II system group OIDs:

Directive	Description
`syslocation`	Used to populate sysLocation OID of MIB-II (1.3.6.1.2.1.1.6.0)
`syscontact`	Used to populate sysContact OID of MIB-II (1.3.6.1.2.1.1.4.0)

Access control information

For BMC Remedy SNMP Agent to respond to user requests, at least one directive specifying access control must be in the `arsnmpd` configuration file. BMC Remedy SNMP Agent supports access control for community-based and user-based access.

Community-based access (described in the following section, "Community-based directives") must be configured for SNMP clients that communicate with BMC Remedy SNMP Agent using either the SNMP v1 or v2c protocol.

User-based access (described in "User-based directives") must be configured for SNMP clients that communicate with BMC Remedy SNMP Agent using the SNMP v3 protocol.

During the installation process, you can configure community-based or user-based authentication, but not both.

In general, because user-based authentication is much more secure than community-based authentication, establishing support for both forms is not recommended. However, if you do enable support for both types of authentication, you must include directives to configure both methods.

Community-based directives

SNMP supports the following categories of communities:

- **Read-only communities**—Have permission to query an SNMP agent for any data that is defined as having read permission. Communities with read-only permission cannot perform SNMP `set` operations that result in a change to the value of a managed object.
- **Read-write communities** —Can view data from an SNMP agent and can change the value of that data (if the OID is defined as having read-write permission) through an SNMP `set` action.

When a client needs to gather information from an SNMP agent that supports community-based authentication, it must supply a plain-text password known as a community string.

To establish a read-only community password, add the following directive:

```
rocommunity communityString
```

To establish a read-write community password, add the following directive:

```
rwcommunity communityString>
```

NOTE

The community string must not include spaces and must not exceed 30 characters in length.

For example:

```
rwcommunity privatecommunity
rocommunity publiccommunity
```

In the previous example, if a client needed to set the value of `arsState` (an action permitted only by those with write permission), it would need to provide the value for the `rwcommunity` directive as part of the SNMP request (`privatecommunity` in this case).

User-based directives

User-based access control is defined in the `arsnmpd` and `snmpd` configuration files. User-based access control defines each user by its level of access, as in community accounts: read-only access or read-write access.

Users can be defined as using one of the following levels of authentication:

- **No authentication and no privacy (noAuthNoPriv)**—Uses no authentication and no privacy functions in the same way as the community-based authentication model. The user name must be supplied to BMC Remedy SNMP Agent, and it functions as a plain-text password (much like a community string). BMC Remedy SNMP Agent does not require a password with a user account configured in this way.

- **Authentication and no privacy (authNoPriv)**—Uses authentication, and no privacy is required to supply a password in addition to a valid user name. BMC Remedy SNMP Agent verifies that the user name and password are correct before acknowledging the client request.
- **Authentication and privacy (authPriv)**—Uses authentication, and privacy is required to supply a password. In addition, the SNMP packet containing the request must be encrypted. BMC Remedy SNMP Agent must have access to the password used by the client to encrypt the packet. It uses this password to decrypt the packet and then verifies that the user name and password are correct.

You define users in the `arsnmpd` file with `rouser` and `rwuser` directives, as follows:

- To create a read-only user account, you must include the following directive:

  ```
  rouser userName [noauth|auth|priv]
  ```

  The optional argument specifies the expected level of encryption to be used by this user. The authentication `noauth` corresponds to `noAuthNoPriv`, `auth` to `authNoPriv`, and `priv` to `authPriv`.

  In the following example, `rouser` directive defines a user account with read-only permission. This account does not require any form of authentication (that is, the user is authenticated in the same way as a user providing a community-string password).

  ```
  rouser user1 noauth
  ```

- To create a read-write user account, you must include the following directive:

  ```
  rwuser userName [noauth|auth|priv]
  ```

  The following example `rwuser` directive defines a user account with read-write permissions. This user must supply a password, but their SNMP requests are not encrypted:

  ```
  rwuser user2 auth
  ```

You can repeat the `rouser` and `rwuser` directives to create multiple user accounts with varying levels of authentication.

The user name supplied to the `rouser` or the `rwuser` directive must not include spaces and must not exceed 30 characters in length.

If the optional argument is not supplied, BMC Remedy SNMP Agent defaults to `auth` level of authentication.

IMPORTANT

The previous directives are not sufficient to properly define a user account. See "The snmpd configuration file" on page 317 for additional configuration requirements.

Trap configuration

Traps are unsolicited messages that BMC Remedy SNMP Agent sends to network management software when unexpected events or errors occur. Messages inform administrators if the AR System process has changed state. Traps can also inform administrators when a client has attempted to access BMC Remedy SNMP Agent using an incorrect community string.

BMC Remedy SNMP Agent can send several standard SNMP traps. Trap messages are formatted using version 1 or version 2 of the SNMP protocol. Using the trap configuration directives, you instruct BMC Remedy SNMP Agent to send a trap to a system that is listening for them on a specific port number.

To send a trap formatted to the SNMP v1 standard, add the following directive to the `arsnmpd` configuration file:

```
trapsink systemNameOrIPAddress communityString [portNumber]
```

To send a trap formatted to the SNMP v2c standard, add the following directive:

```
trap2sink systemNameOrIPAddress communityString [portNumber]
```

You can repeat the trap directives to configure additional systems to receive trap messages.

For example:

```
trapsink traplistener.remedy.com public 8162
```

The preceding directive instructs BMC Remedy SNMP Agent to send trap messages formatted using SNMP v1 to the system `traplistener`, which is listening for trap messages on port number 8162, using community string `public`.

BMC Remedy SNMP Agent can also be configured to send trap messages known as authentication failure traps. These trap messages are sent to all locations specified by the `trapsink`/`trap2sink` directives whenever a client attempts to make an SNMP request using an incorrect community string.

To enable authentication failure trap messages, include the following directive:

```
authtrapenable 1|2
```

Setting `authtrapenable` to `1` instructs BMC Remedy SNMP Agent to send authentication failure traps. Setting the argument to `2` disables this feature.

Location of the armonitor configuration file

To enable BMC Remedy SNMP Agent to interact with AR System, uncomment the following line in the `arsnmpd.cfg` (`arsnmpd.conf`) file:

```
#arsmonitorfile absolutePathToarmonitorFile
```

This is a mandatory configuration directive.

NOTE

Make sure that the argument represents the correct path to your `armonitor` file for your environment.

The snmpd configuration file

The `snmpd` configuration file can contain the following information:

- `engineBoots`
- `engineID`

If an `snmpd` configuration file does not exist, you must create one in the `CONF` directory of your AR System installation. In this case, `engineBoots` and `engineID` is added to the `snmpd` file when BMC Remedy SNMP Agent starts.

In the `snmpd` file, you enter the configuration directives required to fully define a user account. When adding information to this file, do not alter the lines corresponding to `engineBoots` and `engineID` (if they are present).

For each user account defined in the `arsnmpd` file (see `rwuser` and `rouser` directive information in "User-based directives" on page 314), you must include a corresponding `createUser` directive to this file as follows:

```
createUser userName MD5 authenticationPassword DES privatePassword
```

NOTE

Passwords must be at least eight characters long.

Using this directive, you can define the authentication password and privacy password used by the user account (*userName*).

NOTE

In SNMP v3, the authentication password that the user supplies to BMC Remedy SNMP Agent must be encrypted.

The following examples show how directives can be used:

- If you defined a user in the `arsnmpd` file as having read-write permissions and using authentication and no privacy:

  ```
  rwuser user1 auth
  ```

  The following line is required in the `snmpd` file:

  ```
  createUser user1 MD5 mypassword
  ```

- If you defined a user in the `arsnmpd` file as using privacy:

  ```
  rwuser user1 priv
  ```

  The following line is required in the `snmpd` file:

  ```
  createUser user1 MD5 mypassword DES privatepassword
  ```

- If you defined a user in the `arsnmpd` file as using no authentication and no privacy:

  ```
  rwuser user1 noauth
  ```

  The following line is required in the `snmpd` file:

  ```
  createUser user1
  ```

The armonitor configuration file

The `armonitor` configuration file permits the `armonitor` utility to start BMC Remedy SNMP Agent and to establish a link to it.

If you configured BMC Remedy SNMP Agent during installation, no modification to this file is necessary. If you did not configure BMC Remedy SNMP Agent during installation, you must edit the `armonitor.cfg (armonitor.conf)` file to enable `armonitor` to start and interact with BMC Remedy SNMP Agent.

Enable SNMP by setting the configuration parameter SNMP-agent-enabled to true as follows:

```
SNMP-agent-enabled: T
```

In addition, remove the comment marker (#) from the command line corresponding to the `arsnmpd` process. (This enables `armonitor` to start BMC Remedy SNMP Agent.)

You might need to change the default port number from 161 because an SNMP agent might already be running on the default port 161. To do this, change the value of `upd:161` on the line corresponding to the `arsnmpd` process to a new port number that is not currently in use by another process, for example `upd:8161`.

Starting BMC Remedy SNMP Agent

You can start BMC Remedy SNMP Agent in any of the following ways:

- Using `armonitor`
  - If you configured SNMP during installation, `armonitor` with start the SNMP agent automatically after installation is complete.
  - If the SNMP agent process is terminated, `armonitor` restarts the SNMP agent.
- Using a command line with the following syntax:

  ```
  arsnmpd -c pathToarsnmpdConfigurationFile udp:portNumber
  ```

  Make sure that:
  - The argument to the `-c` option is the path to the `arsnmpd` configuration file, not the `snmpd` file.
  - The argument to `udp` is a port number not currently in use by another SNMP agent or any other process.

    For example (UNIX):

    ```
    arsnmpd -c /user/ar/arsystem/conf/arsnmpd.conf udp:8161
    ```
- Invoking the agent during AR System startup, using the Services panel (on Windows) or the `arsystem` shell script (on UNIX).

Stopping BMC Remedy SNMP Agent

On Windows, AR System is installed as a service. If you stop the AR System service, BMC Remedy SNMP Agent also stops.

On UNIX, use the `arsystem` shell script to stop AR System, which stops all AR System processes, including BMC Remedy SNMP Agent.

If you use BMC Remedy SNMP Agent to stop AR System, BMC Remedy SNMP Agent exists as an independent process that is not under the control of the `armonitor`, the AR System service (Windows), or the `arsystem` shell script (UNIX). You must manually stop and restart BMC Remedy SNMP Agent by using specific operating system methods (for example, by using Task Manager on Windows or by using a `kill` command on UNIX).

To stop BMC Remedy SNMP Agent without affecting other AR System processes, use standard operating system approaches to stopping individual processes. (Often this can be accomplished on either a Windows or UNIX system by issuing a `kill` command from a command prompt.) If BMC Remedy SNMP Agent is still a child process of `armonitor`, however, `armonitor` attempts to restart the agent.

For more information about stopping individual processes, see your system administrator.

Troubleshooting

This section describes some issues you might encounter with SNMP, and possible resolutions.

Problem or question	Resolution
You configured BMC Remedy SNMP Agent, but it does not start.	BMC Remedy SNMP Agent might be using a port number already in use. SNMP agents are often present and running on many operating systems (especially UNIX) using the default port 161. Open the `armonitor.cfg` (`armonitor.conf`) file and change the port number used by BMC Remedy SNMP Agent. Restart AR System.
You instructed BMC Remedy SNMP Agent to change the value of `arsState`, but it does not respond to requests. All requests time out.	When the value of `arsState` is changed (to 1 or 2) by an SNMP `set` request, BMC Remedy SNMP Agent interprets this as a command to alter the state of the `arsSystem` to match the new value. It does not respond to additional SNMP requests until either of the following events occur: ▪ The call to stop AR System returns. ▪ Eight minutes elapse (if attempting to set `arsState` to 1). Make sure that you supplied a community string or a user name that has write permissions.
When you query for the current value of `arsState`, the result is 2 (down) even though the server is running.	Check the value of `SNMP-agent-enabled` in the `armonitor.cfg` (`armonitor.conf`) file . The value might be set to `F`. If so, set it to `T` and restart AR System.

Problem or question	Resolution
You set the value of `arsState`, but there is no change in the state of AR System.	Check the value of `SNMP-agent-enabled` in the `armonitor.cfg` (`armonitor.conf`) file. If this is set to `F`, you cannot stop or restart AR System. Set this value to `T` and restart AR System. If the value is already set to `T`, verify the following information: ■ The `arsnmpd` process is running on the system with which you are trying to communicate. ■ You supplied the community string (or user name) that permits read-write operations.
When you make a change to your configuration file, do you need to restart the `arsnmpd` process?	You do not need to restart the `arsnmpd` process. If you perform an SNMP `set` operation to change the value of `versionUpdateConfig.0` (.1.3.6.1.4.1.2021.100.11.0) to 1, BMC Remedy SNMP Agent rereads the `arsnmpd.cfg` (`arsnmpd.conf`) file.
You want to monitor IP traffic from the MIB-II group, but BMC Remedy SNMP Agent does not respond. Do all SNMP agents support MIB-II?	BMC Remedy SNMP Agent supports only the system and SNMP portions of MIB-II at this time. To gather additional MIB-II data, query the SNMP agent that is monitoring your operating system.

Problem or question	Resolution
You configured BMC Remedy SNMP Agent to send trap messages to your Network Management Station (NMS), but there are no messages.	Verify the following information: ■ Check the `arsnmpd.log` file for trap-related error messages. If BMC Remedy SNMP Agent cannot connect to a trap receiver, you see the following entry: `Error: Cannot create trapsink: nameOfTrapReceiver.` ■ The agent is using the correct port number for sending traps to your Network Management Station (NMS). Most NMSs listen to the default port of 162 for receiving SNMP trap messages. ■ You supplied the correct community string for your NMS in the `trapsink/trapsink` directive in the `arsnmpd.cfg` (`arsnmpd.conf`) file. ■ The NMS supports the type of trap message you are sending. Traps are formatted according to either v1 or v2 of the SNMP protocol. If you configured BMC Remedy SNMP Agent to send trap messages formatted according to v2 to an NMS that supports only v1, you must update your BMC Remedy SNMP Agent configuration to send v1 traps instead. ■ Any daemon processes required for receiving trap messages are running on your NMS.
BMC Remedy SNMP Agent is running, but you do not receive any information. All requests time out.	Verify the following information: ■ You are using proper authentication. If you provided an incorrect user name or password, your request times out because BMC Remedy SNMP Agent does not respond. ■ You configured all access control components. (BMC Remedy SNMP Agent can run without an `arsnmpd` configuration file.) Check the `arsnmpd.log` file in the `db` directory of your AR System installation. If you did not configure access control in the `arsnmpd` configuration file, you see the following entry: `Warning: no access control information configured. Remedy SNMP agent cannot function in this state.`

Chapter

23 Making applications licensable for integration system vendors

When creating AR System applications, integration system vendors (ISVs) authorized by BMC can make the applications licensable. This section describes how to make your applications licensable.

The following topics are provided:

- Application licensing options (page 324)
- Application licensing overview (page 325)
- Making applications licensable (page 326)
- Adding the application license to your server (page 327)
- Assigning application licenses to users (page 328)

NOTE

The procedures in this section are intended for use *only* by authorized ISVs who plan to license their applications for sale. If you are not an ISV and you create BMC Remedy applications that you want to license for sale, see the BMC Customer Support website (`http://www.bmc.com/support`) for details.

Application licensing options

When licensing AR System applications, you have two options:

- **Application licensable**—Users must obtain an "application license" so that they can access the form data in the application. Without a valid application license installed on the server, users receive a licensing error when they try to perform any data-related operations on forms in the unlicensed application. In other words, they cannot get, modify, search, create, delete, or merge entries on any form.
- **Application *and* user licensable**—In addition to the requirements for the application licensable option, users must obtain user-fixed or user-floating licenses based on individual forms. The application developer can choose which forms in the application to make "user licensable," and users must have an application user license to access form data on these forms. These are the application user license types:
  - Fixed
  - Floating
  - Read

  Fixed and floating licenses have no restrictions and enable users to get, modify, search, create, delete, and merge entries. Read licenses enable users only to search, get, and create entries.

Operation on entries	License requirements
`Create`	Does not need application or application user license.
`Get`	Needs application license only.
`Modify`	Needs application and application user license.
`Delete`	Needs application and application user license.
`Merge`	■ If a merge operation results in a *create*, does not need application or application user license. ■ If a merge operation results in a *modify*, needs application and application user license.

By default, an application comes with zero fixed licenses, zero floating licenses, and an unlimited number of read licenses. If users are not assigned a fixed or floating license, they automatically use a read license by default. Guest users automatically use a read license.

NOTE

This feature is intended to license your access to form data in the application; it is *not* intended to license administrative operations such as modifying forms or workflow in an application. Even if no application license is installed for a licensable application, administrators can still change or delete a form or workflow object.

Application licensing overview

In this example, XYZ Corporation, an ISV, created an application, MusicManager, that they want to license for sale. The following steps describe the process to license their application. Typically, ISVs perform step 1 through step 3. Customers who purchase the application perform steps 4 and 5.

Step 1 Create the MusicManager application with its accompanying forms.

Step 2 Register your application name with BMC.

BMC applications *must* have unique names. To avoid potential licensing conflicts, use this naming convention:

`vendorName:applicationName`

In this example, the name is XYZ:MusicManager.

Usually, you do not need to worry about conflicts with naming conventions for your application. Even if XYZ has a competitor with its own MusicManager application, the vendor name prefix guarantees uniqueness in most cases.

If you have questions, go to the Customer Support website (`http://www.bmc.com/support`) and verify that the application name is unique or has not been used by another application developer in your company.

Step 3 Designate the application as licensable.

You can make your application *application licensable* or *user licensable*, as described in "Application licensing options" on page 324:

- Customers need application licenses to access the form data in the application.
- Customers need user fixed or user floating licenses based on individual forms to access form data.

In this example, XYZ Corporation makes the MusicManager application licensable *and* user licensable. The MusicManager application includes these forms: MusicManager Configuration, MusicManager Songs, and MusicManager Singers. When customers try to access data in any of these forms, they receive an error if they do not have a valid MusicManager application license.

Further, XYZ Corporation makes *only* the MusicManager Songs form user licensable (shown in Figure 23-1). If users try to submit or modify data in this form, they receive an error if they do not have a MusicManager user license (fixed or floating). Because the ISV does not make the MusicManager Singers form user licensable, any user can create or modify MusicManager Singers without a MusicManager user license.

For detailed steps, see "Making applications licensable" on page 326.

Step 4 (Customers only) In the AR System Administration Console, click System > General > Add or Remove Licenses to add the license to your AR System server.

For information, see "Adding the application license to your server" on page 327.

Step 5 (Customers only) If your application is user licensable, assign application user licenses to your users so that they can access the user licensable portions of the application (the user licensable forms).

To do this, open the User form in the AR System Administration Console and assign licenses to specific users. After users are given the appropriate application user licenses, they can access the application and its forms in the usual manner. For information, see "Assigning application licenses to users" on page 328.

Making applications licensable

ISVs use the following procedure to make their applications licensable. In this example, the MusicManager application contains three forms, but you want to make only one of them (MusicManager Songs) user licensable.

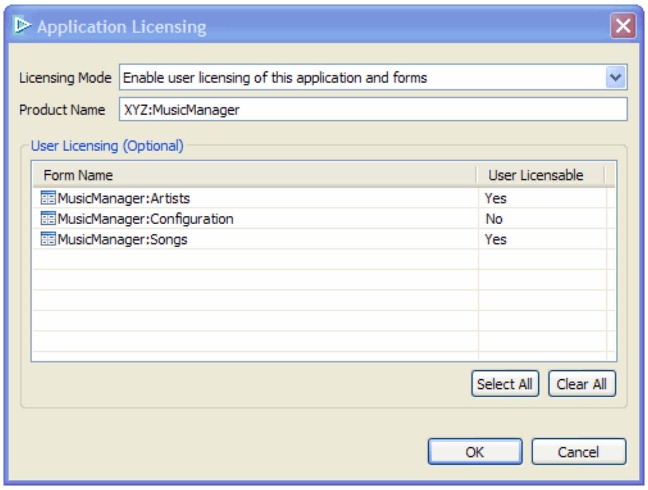
WARNING

Licensing mode is a one-way operation. After an application is made licensable, the process cannot be reversed, even by the ISV. In addition, after an application is user licensed and the forms are checked, the forms cannot be unchecked and you cannot revert to licensing or nonlicensing of the application. Export copies of these forms *before* you license the application so that you can recover the forms if you need to change licensing levels later on.

▶ To configure your applications to make them licensable

1 Choose Application > License Application.

Figure 23-1: Application Licensing dialog box



Licensing Mode: Enable user licensing of this application and forms
Product Name: XYZ:MusicManager

Form Name	User Licensable
MusicManager:Artists	Yes
MusicManager:Configuration	No
MusicManager:Songs	Yes

2 From the Licensing Mode menu, select one of the following options:

Option	Description
Do not license this application	Application is not licensed. Typically, you use this default option if you do not intend to sell this application or if the application is for internal use only.
License this application	Entire application is licensed. An application license must be added to the Add or Remove Licenses form for users to access the form data in the application.
Enable user licensing of this application and forms	Specific forms in the applications can be made user-licensable. You decide which forms to make licensable. Users must obtain a user license to create, modify, merge, or delete form data. These licenses can be fixed, floating, or read. Note: Users have read license permission if they are not assigned fixed or floating licenses. By choosing which forms in your application to make user-licensable, you can customize the user licensability of your application.

3 In the Product Name field, enter the application license string, for example, XYZ:MusicManager.

4 If you enable user licensing for the application and its forms, perform one of these actions:

- Click inside the User Licensable list adjacent to the form you want to license, then choose Yes or No. Repeat for other forms.
- To license *all* forms in your application, click Select All.
- To clear all forms from the User Licensable list, click Clear All.

This step is optional for applications and is needed only to make forms in your application user licensable.

5 Click OK.

Adding the application license to your server

Customers license ISV applications just like any other AR System application. The only difference is the unique naming convention of the application itself, indicating that the application comes from an ISV.

For information about adding application licenses to servers, see the *Configuration Guide*.

Assigning application licenses to users

After adding your application licenses to the server, follow this procedure to assign licenses for your application to users.

▶ To assign licenses to users

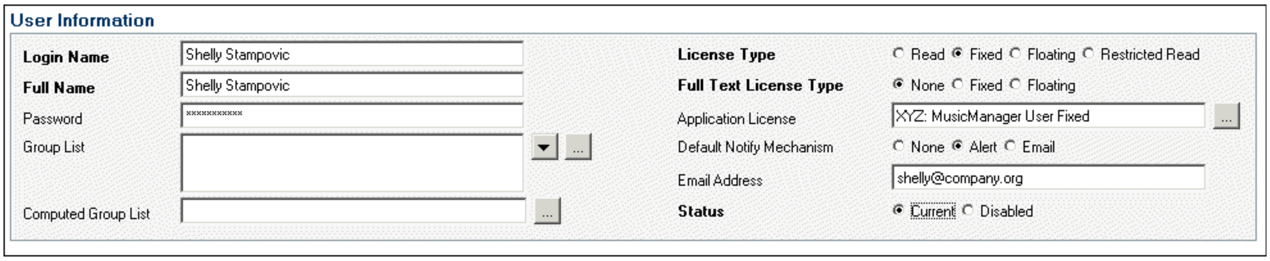
1 In the AR System Administration Console, click Application > Users / Groups / Roles > Users.

2 In the User form, create a user (New mode) or find an existing user (Search mode).

3 Enter information in the appropriate fields to create or modify users.

4 In the Application License field, enter the name of the application and the appropriate license type, as shown in Figure 23-2.

Figure 23-2: Entering application license information on the User form



User Information

Login Name	Shelly Stampovic	**License Type**	Read [✓ Fixed] Floating Restricted Read
Full Name	Shelly Stampovic	**Full Text License Type**	[✓ None] Fixed Floating
Password	**********	Application License	XYZ: MusicManager User Fixed
Group List		Default Notify Mechanism	None [✓ Alert] Email
		Email Address	shelly@company.org
Computed Group List		**Status**	[✓ Current] Disabled

NOTE

Use the Application License field to provide fixed or floating application licenses to users. The License Type field is applicable only for BMC Remedy AR System user licenses and *not* for application licenses. To provide a user with write access to the system, set the License Type field to Fixed or Floating.

Use the following syntax when providing users with application licenses:

```
vendorName:applicationName user typeOfLicense
XYZ:MusicManager User Fixed
```

Separate multiple licenses with semicolons:

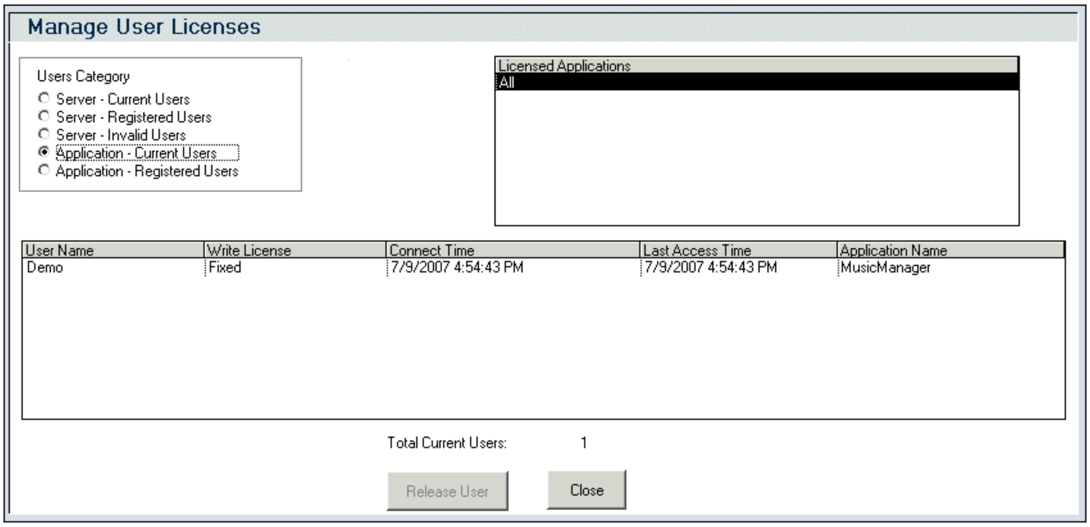
```
XYZ:MusicManager User Fixed; XYZ:NoiseManager User Fixed
```

5 Save your changes.

In this example, one fixed user license was issued so that the user could search or modify the form data of the `XYZ:MusicManager` application.

These application licenses function like other user licenses for BMC Remedy products. For example, you can view current user information in the Manage User Licenses window in the AR System Administration Console, as shown in Figure 23-3.

Figure 23-3: License information



Application - Current Users [✓]

Licensed Applications
All

User Name	Write License	Connect Time	Last Access Time	Application Name
Demo	Fixed	7/9/2007 4:54:43 PM	7/9/2007 4:54:43 PM	MusicManager

Total Current Users: 1

For more information about managing licenses, see the *Configuration Guide*.

▶ To assign licenses to license pools

1 In the Group form, create a new group (New mode) or find an existing group (Search mode).

2 Enter the required information in the appropriate fields to create or modify groups.

3 Enter the application name and the number of application floating licenses to be held in the application license pool for a particular group in the Application Floating License field.

NOTE

Users who need to be granted these application floating licences must be a part of this group.

For information about the syntax of the Application Floating License field, see the *Form and Application Objects Guide*, "Creating and managing groups," page 49.

4 Save the changes.

The application license pool is created.

Appendix

A Web service operation types

This section describes the types of web service operations that can be used with AR System.

The following topics are provided:

- Create operation type (page 332)
- Set operation type (page 332)
- Get operation type (page 333)
- Service operation type (page 334)
- XPATH function (page 335)
- Setting the start record and the maximum limit (page 337)

For additional information about web services, see Chapter 6, “Web services.”

Create operation type

External applications can use a `Create` operation to submit new entries into AR System.

When AR System receives an incoming `Create` request, it performs these actions:

- Parses the incoming XML code based on the input mapping, and generates field values.
- Creates an entry in the base form with these field values.

  The new entry creation triggers the `OnSubmit` filters.
- After the `Create` action is completely successful, obtains field values from the newly created entry.
- Uses the output mapping to generate XML code from these field values, and sends the XML document back as a response.

The output mapping might be different from the input mapping. The input mapping has the incoming data, but the output mapping has computed data, for example the `EntryId`, the `CreateDate`, or any fields that are set by filters.

The `Create` operation is similar to the action of filling in a form, submitting it, and then searching for the same entry. All the rules for `Submit` also apply—required values must be entered, system fields cannot be submitted, and so on.

A `Create` operation can create only one entry in the base form. You can add it to a web service multiple times, each time specifying a unique name for the operation and a set of input and output mappings. Multiple `Create` operations are useful to make different operations available on the same base form to accommodate different connecting applications.

Set operation type

External applications can use a `Set` operation to modify an existing entry in AR System.

Unlike the `Create` operation, `Set` needs a qualification to identify the entry to modify. This qualification must be specified in BMC Remedy Developer Studio at design time. You cannot use an attachment field as a field reference in a qualification. The qualification might be completely static, for example `‘RequestID’ = “000000000000001”`, which means that incoming requests always operate on the first entry.

A more useful qualification is either semidynamic or completely dynamic.

- A semidynamic qualification looks like a static qualification except that its values can be `XPATH` expressions, for example, `‘RequestID’ = XPATH(“/ROOT/RequestID”)`.
- A completely dynamic qualification is a single `XPATH` expression, for example, `XPATH(“/ROOT/QueryString”)`.

For information about XPATH expressions, see "XPATH function" on page 335.

When AR System receives an incoming Set request, it performs these actions:

- Determines whether the qualification is dynamic; if so, it expands the XPATH expressions with data from the incoming XML to make the qualification static.
- Searches the base form for an entry matching the qualification.
- Parses the incoming XML code based on the input mapping, generates field values, and modifies the matched entry.

  The entry modification triggers the OnModify filters.
- After the modify action is successfully completed, obtains field values from the recently modified entry.

  The action of obtaining the field values triggers the OnGetEntry filters.
- Uses the output mapping to generate XML code from these field values, and sends the XML document back as a response.

As in the Create operation, AR System returns the same entry as the one submitted because the output mapping might be different from the input mapping. The input mapping has the incoming data, but the output mapping has computed data, for example fields that are set by filters.

The Set operation is similar to modifying an entry in BMC Remedy User and then clicking refresh to get back the same entry. All the rules for modify also apply—required values cannot be nulled out, and system fields cannot be changed.

The Set operation can modify only one entry in the base form. You can add it to a web service multiple times, each time specifying a unique name for the operation and a set of input and output mappings.

See "The Set operation type for complex documents" on page 344.

Get operation type

External applications can use a Get operation to get details about an entry in AR System.

Like the Set operation, Get needs a qualification that must be specified in BMC Remedy Developer Studio. You cannot use an attachment field as a field reference in a qualification. For the Get operation, a qualification is automatically generated for you, and a particular entry is retrieved. The qualification can be modified. For the GetList operation, you must manually enter a qualification, or the system retrieves all the entries.

Qualifications for a Get or GetList operation can be static, semidynamic, or completely dynamic. These qualification types are described in "Set operation type" on page 332.

When AR System receives an incoming `Get` request, it performs these actions:

- Determines whether the qualification is dynamic. If so, it expands the `XPATH` expressions with data from the incoming XML to make the qualification static.
- Searches the base form for entries matching this qualification. Unlike `Set` and `Create`, `Get` can handle multiple entries. Also unlike `Set` and `Create`, AR System completely ignores the input mapping; the input XML is used only for expanding `XPATH` expressions. For information about `XPATH` expressions, see "XPATH function" on page 335.
- Gets field values from the matched entries.

  The action of obtaining the field values triggers the `OnGetEntry` filters.
- Uses the output mapping to generate XML code from these field values, and sends an XML document back as a response.

You do not need an input mapping for the `Get` operation; you can create it, but the system ignores it.

The default operations listed are `Get` and `GetList`. These are merely names for the operation type `Get`. Use the mappings to create a `Get` or a `GetList` operation. `Get` returns one entry, and `GetList` returns multiple entries. For `GetList` operations, map the form to a complex type with `maxOccurs=`*`aNumberGreaterThan1orUnbounded`* so that the resulting records (>1) can be passed to the user. For more information, see "The Get operation type for complex documents" on page 345.

Service operation type

External applications can use the Service operation to execute the Service Entry filters on a form. For more details on Service Entry see the *Workflow Objects Guide* and the *C API Reference*.

This operation type does not work on complex documents, so the document can have only one entry for one form.

When AR System receives an incoming Service request, it performs these actions:

- If the Primary Key is not Request ID, determines the entry ID for the item.
- Executes the filters associated with the Service Execution Option, associated with the form.
- Ensures that the fields defined in the input mapping are the input fields to the Service Entry call and the fields defined in the output mapping are the fields expected in the output field list.
- Takes the output field list and converts it into an XML document and returns it.

The default operation is listed as `Service`. These are merely names for the operation type Service. Use the mappings to create the input field list and output field list.

XPATH function

When web services are published in AR System, the `Get` and `Set` operation types accept a qualification string. At run time, a query is made using the specified qualification, and the `Get`, `GetList`, and `Set` operations are executed on the entries returned by the query. The format of the qualification string is similar to that used in the advanced query bar in the user client, but an additional function called `XPATH` is used for web service qualifications.

To access the available `XPATH` expressions, use the Expression Editor. See "To construct XPATH expressions in the Expression Editor" on page 336.

An `XPATH` expression identifies an XML element inside an XML document. Its syntax is similar to a directory path. When creating `XPATH` expressions, follow these guidelines:

- The `XPATH` function takes one argument, which must be an expression referencing an element or an attribute in the input mapping of the operation.
- The element or attribute being referenced does not have to be mapped to a field; it might exist only in the XML schema and be used only for qualification.
- The expression must start with /ROOT and must list all the elements in the path including the one being referenced.
- When referencing an attribute, use @ before the attribute name.
- The `XPATH` function can be used anywhere in the qualification, and a value is substituted based on the XML data type.
- For strings, AR System adds extra double quotation marks around the value.
- For date-time fields, the XML date time string is converted to the number of seconds.
- If more than one element matches the expression, the first element is considered.
- An entire qualification string can be passed as one of the XML elements in the input document to create totally dynamic queries. This is analogous to using the EXTERNAL function.
- If an element is missing in the input document, it is resolved to $NULL$.

`XPATH` expressions are similar to field references, for example, suppose you have this qualification:

```
'RequestID' = $RequestID$
```

`$RequestID$` is the value of the RequestID field in the current entry, and `'RequestID'` is the field to search on.

Similarly:

```
'RequestID' = XPATH("/ROOT/RequestID")
```

`XPATH("/ROOT/RequestID")` is the value of the RequestID in the current XML document, and `'RequestID'` is the field to search on.

This is an example of an XML document and some sample qualification strings using XPATH expressions:

```
<? xml version=”1.0” ?>
<ROOT>
  <Employee ID=”112”>Adam</Employee>
  <Address>
    <Street>1500 Salado Dr</Street>
    <City>Mountain View</City>
  </Address>
  <HireDate>2004-01-01T00:00:00.0000000-08:00</HireDate>
  <query>
    <qualification>’Employee_ID’ > 100</qualification>
  </query>
</ROOT>
```

Sample qualification	XPATH resolved qualification	Comments
`‘Employee_Name’=XPATH(/ROOT/Employee)`	`‘Employee_Name’=”Adam”`	Employee is of type string.
`‘Employee_ID’=XPATH(/ROOT/Employee/@ID)`	`‘Employee_ID’=112`	ID is an attribute of type integer.
`‘City’=XPATH(/ROOT/Address/City)`	`‘City’=”Mountain View”`	City is of type string.
`‘HireDate’=XPATH(/ROOT/HireDate)`	`‘HireDate’=1009872000`	HireDate is of type dateTime. It converts to the number of seconds since 1/1/1970.
`‘State’=XPATH(/ROOT/Address/State)`	`‘State’=$NULL$`	State does not exist in the input document.
`XPATH(/ROOT/query/qualification)`	`‘Employee_ID’>100`	Qualification is of type string.

For information about the XPATH specification, go to http://www.w3.org/TR/xpath.

▶ To construct XPATH expressions in the Expression Editor

1. Expand the WSDL operation panel where you want to add the XPATH expression.
2. Click the ellipsis button [...] adjacent to the Qualification field to open the Expression Editor.
3. In the Expression Editor, expand the Fields panel.
4. Double-click a field name, so that it appears in the editing field.
5. Click the = button.
6. Expand the XPATH panel.
7. Double-click an XML element, so that it appears in the expression editing area.

   Alternatively, you can highlight an element and click the Add XPATH button.
8. Verify that your XPATH expression is correct—for example:

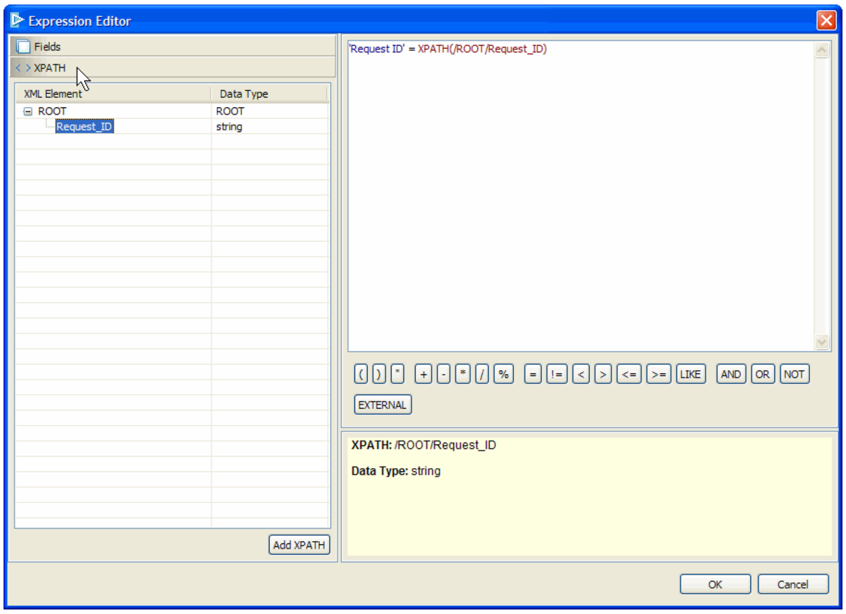
   ```
   'Request ID' = XPATH(/ROOT/Request_ID)
   ```

9 Click OK to save your qualification.

TIP

You can also construct expressions by typing in the editing field.

Figure A-1: XPATH panel in the Expression Editor



XML Element	Data Type
ROOT	ROOT
Request_ID	string

'Request ID' = XPATH(/ROOT/Request_ID)

XPATH: /ROOT/Request_ID

Data Type: string

For more information about using the Expression Editor, see the *Workflow Objects Guide*.

Setting the start record and the maximum limit

If you select the `GetList` operation, an `XPATH` expression is displayed in the Start Record field. This expression determines the zero-indexed first element returned when the data is fetched. For example, to start at the first record matching the qualification, use a value of 0. Similarly, the Max Limit field has an `XPATH` expression that retrieves the value for the maximum number of records to return from the XML document.

For example, suppose the Max Limit expression determines that 10 records should be shown at a time. If 10,000 records match the qualification, only 10 records starting from the Start Record are displayed.

Appendix

B Mapping web service data

This section describes the tools and processes used to map web service data.

The following topics are provided:

- Mapping to simple and complex documents (page 340)
- XML editing (page 351)
- Data types (page 360)

For additional information about web services, see Chapter 6, “Web services.”

Mapping to simple and complex documents

The procedure for mapping documents is the same for both creating and consuming web services.

AR System performs default mapping automatically when you create a web service, or when you add an operation to a web service. By default, your base form is mapped to the ROOT element in your XML Schema, and the fields in your form are automatically mapped to element names in the XML document.

Simple documents

With a simple XML schema, the fields from one form are mapped to the XML element names. The map is displayed as a list of ARFieldName/XMLElementName pairs.

▶ To map to simple documents

1. In BMC Remedy Developer Studio, open a web service for editing.
2. In the web services editor, expand the WSDL Ports panel, the Port panel, and the WSDL Operations panel.
3. Click on the operation you want to map, expanding the corresponding panel.

   Tables for input mapping and output mapping appear.
4. Set the Data Source Type to Generated.

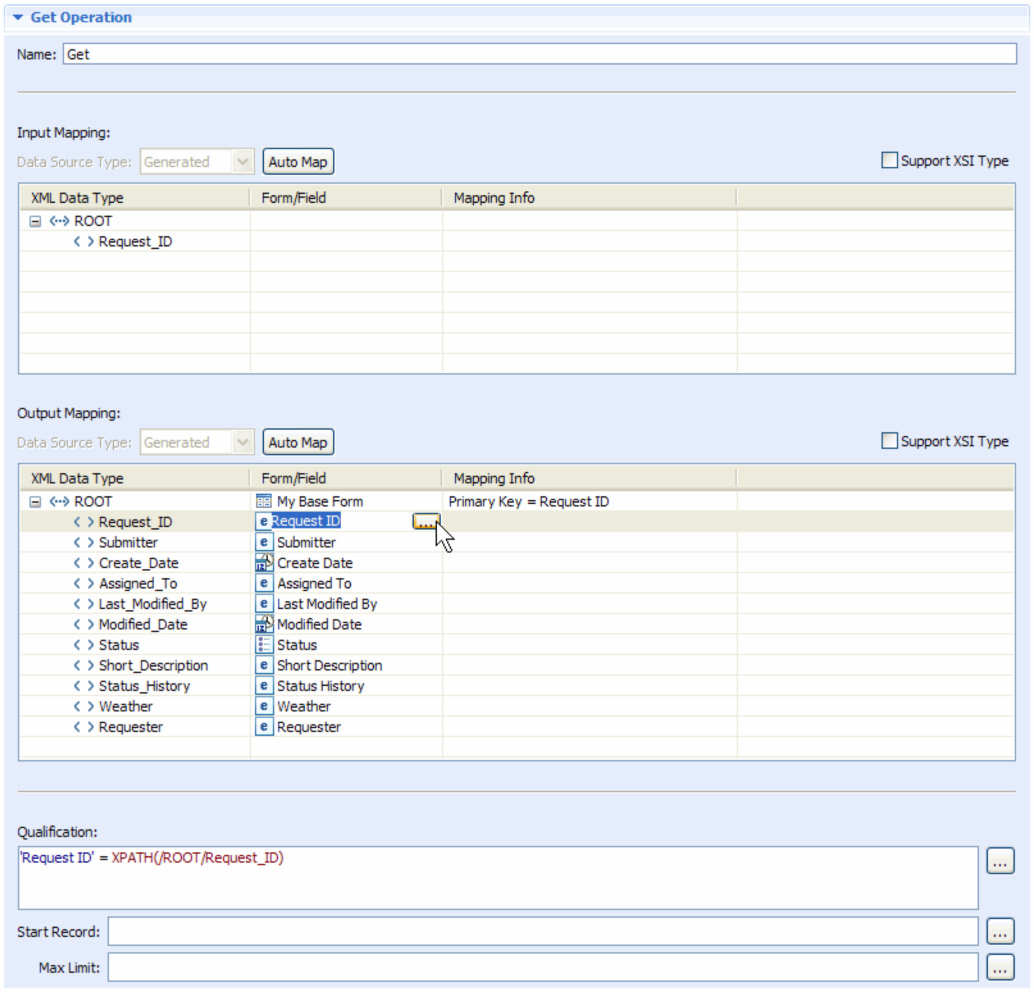
   If you did not specify an XML Schema for the web service, the Data Source Type is automatically set to Generated.
5. Click the Auto Map button to regenerate default mappings.

   The XML element names are displayed in the XML Data Type column. The base form name and fields are displayed in the Form/Field column.

NOTE

AR System sets default mappings automatically when you create a web service, or when you add an operation to a web service. If the existing mappings are satisfactory, you do not need to regenerate them.

Figure B-1: Mapping for a Get operation (simple document)



Get Operation

Name: Get

Input Mapping:

Data Source Type: Generated

XML Data Type	Form/Field	Mapping Info
ROOT		
Request_ID		

Output Mapping:

Data Source Type: Generated

XML Data Type	Form/Field	Mapping Info
ROOT	My Base Form	Primary Key = Request ID
Request_ID	Request ID	
Submitter	Submitter	
Create_Date	Create Date	
Assigned_To	Assigned To	
Last_Modified_By	Last Modified By	
Modified_Date	Modified Date	
Status	Status	
Short_Description	Short Description	
Status_History	Status History	
Weather	Weather	
Requester	Requester	

Qualification: 'Request ID' = XPATH(/ROOT/Request_ID)

Start Record:

Max Limit:

6 Review the mappings, and make any necessary adjustments to forms, fields, and elements in your XML schema. (See Figure B-1.)

To modify the properties of an object in the XML Data Type column, select the object and click the adjacent ellipsis button [...]. To view XML properties, hover your mouse over the object.

To replace a form or field in the Form / Field column, select the object and click the adjacent ellipsis button [...].

Mapped items must be of the same data type (see “Data types” on page 360).

7 Make any adjustments to the mapping, or to the form, fields, and elements in your XML schema.

8 Save your web service.

To remove existing mappings

1. Right-click a row in the Input Mapping or Output Mapping table.
2. Click Remove Selected.

> ***TIP***
>
> If you remove a mapping by mistake, you can easily re-create it. Right-click the ROOT element in the XML Data Type column, choose New > Element > Field, rename the new field element, then map it to the appropriate field.

3. Save your web service.

To map a field to an XML element name

1. In the Input Mapping or Output Mapping table, locate the XML element to which you want to map a field, and highlight the corresponding row.
2. In the highlighted row, click the current field name that is mapped to your XML element.

   An ellipsis button [...] appears adjacent to the field name.
3. Click the ellipsis button [...], then select the field you want to map and click OK.
4. Save your web service.

   To add fields or make other modifications to the XML schema, see "To add a new element or attribute" on page 352 and "To change element ordering" on page 352. To remove fields, see "To remove an existing field from the mapping list" on page 357.

Complex hierarchical documents

For complex hierarchical documents, the XML schema maps to multiple interrelated forms. In the following example, a purchase order XML document is mapped to two forms, Purchase Order and Line Item, as represented by this figure:

Purchase Order		
Request ID	**Description**	**Date**
007	XYZ Corp	2/12/08
012	ABC, Inc.	3/01/08

Line Item			
Request ID	**Line ID**	**Description**	**PO ID**
0015	1	Memory	007
0016	2	CPU	007
0017	3	HardDisk	007
0020	1	Scanner	012
0021	2	Printer	012

The data consists of two purchase orders, one for XYZ Corporation with three line items (Memory, CPU, and HardDisk) and one for ABC, Inc. with two line items (Scanner and Printer). The Purchase Order and Line Item forms are related through their ID fields:

- Request ID on the Purchase Order form. This is the primary key in the parent form and it is unique. This is the key that establishes the relationship with the foreign key in the LineItem form.
- PO ID on the Line Item form. This is the foreign key in the child form; it establishes the relationship primary key in the Purchase Order form.
- Request ID on the Line Item form. This is the primary key in the child form, and it is unique.
- Line ID on the Line Item form. This is unique only in the subset of requests that reference the same Purchase Order form. The Line ID together with the PO ID form a "unique key."

The XML input document in the example can be represented as follows:

```
<PurchaseOrder>
  <Customer>XYZ Corp</Customer>
  <Date>2/12/04</Date>
  <Items>
    <LineItem> <Id>1</Id>  <Description>Memory</Description>
    </LineItem>
    <LineItem> <Id>2</Id>  <Description>CPU</Description>
    </LineItem>
    <LineItem> <Id>3</Id>  <Description>HardDisk</Description>
    </LineItem>
  </Items>
</PurchaseOrder>
```

The XML document does not include Request IDs. Request IDs have no meaning outside AR System unless they are used as the primary key identifier for the document. For example, if the purchase order (PO) document uses the Request ID field as the PO Number, that number is also used externally. In this case, the request ID field is probably renamed as the PO Number field. The server automatically creates Request IDs for the parent and child forms, and assigns foreign keys to the child form as the identifier between the child and parent.

The only IDs in the XML document are the Line IDs of the child form. Line IDs can be numbers or strings, such as a description. Line IDs are used only in the modify operation: the server compares the existing complex document with the new document and determines which child items to modify, insert, and delete.

Set operations with line items

For a complex document, for example, a purchase order with three line items, where you submit an XML document containing two line items for a Purchase Order already existing in the system, the server compares the Line IDs of the existing three line items with the Line IDs of the two new line items. The following rules apply:

- If there are matching Line IDs, the server updates the line item in the database with the contents of the XML elements inside the corresponding line item in the input document. Fields for which the XML elements are missing are left untouched.
- If a new Line ID does not exist in the database, the server creates a record in the line item form.
- If there is a missing Line ID—that is, if a line item exists in the server but not in the input request—the server either deletes the line item from the server or ignores it, depending on the option you select in BMC Remedy Developer Studio on the Set Operation panel.
  - If you select Full from the list in the Composite Options field, the server deletes missing line items.
  - If you select Partial from the list in the Composite Options field, the server ignores missing line items.

As a consequence of these rules, a modify operation for a complex document can result in create and delete actions.

Make sure that each Line ID is unique within the scope of one document. The server cannot distinguish between duplicate LineIDs when performing a modify action. The use of duplicate Line IDs might cause unexpected results.

The Set operation type for complex documents

The `Set` (or `Modify`) operation for complex documents is more complicated than it is for simple documents. In a simple document, the base form might have ten fields, for example, all mapped to XML elements. If the XML for your modify request is returned with all ten XML elements, the server modifies each of the ten fields with data from the ten XML elements. However, if your modify-request XML is returned with fewer than ten elements and you specified `MinOccurs=0` in the Object Properties dialog box for the XML elements, the server modifies only the fields in the input request. If your modify-request XML is returned with fewer than ten elements and you set `MinOccurs` to a value other than 0, you receive an error message.

The Get operation type for complex documents

For `GetList` operations, map the form to a complex type and set `maxOccurs` to a number greater than 1 or `unbounded`, so that the resulting records (>1) can be passed to the user. Instead of mapping the form to the ROOT element, which cannot have `maxOccurs=unbounded`, map to another element below ROOT which has `maxOccurs=unbounded`. All the fields on the form for a `GetAll` operation should be mapped to the children of this element rather than being mapped to children of ROOT. The default `GetList` operation already has this element called `getListValues`; however, if you are creating custom mappings, make sure that this element exists.

NOTE

The `GetList` operation uses more system resources than a native API call does when retrieving data from AR System. To improve performance when using the `GetList` operation, retrieve the data in smaller chunks. See "Setting the start record and the maximum limit" on page 337.

Filter flow for complex documents

When a complex document such as PurchaseOrder (described on page 342) is received by the AR System server, the AR System server updates the Line Item form by generating Push Fields actions on the Purchase Order form for every line item in the PurchaseOrder document.

- For publishing—The Push Fields actions relating to the web service are executed before other Push Fields actions on the Purchase Order form are executed.
- For consuming—The Push Fields actions generated during the consumption of a web service are executed before other Push Fields actions on the Purchase Order form.
- For a form with both publishing and consuming web services—The filter flow is as follows:

  Publishing Push Fields actions > Consuming Push Fields actions > Other Push Fields actions.

Mapping to complex documents

To create the mapping of a complex document, you first need to create forms and a complex XML Schema. See "Advanced XML editing" on page 358. Choose the parent form as the base form of the web service. Additional forms are used as child forms. You should not use a child form with more than one parent form.

▶ To map to complex documents

1 In BMC Remedy Developer Studio, open a web service for editing.

2 In the web services editor, expand the WSDL Ports panel, the Port panel, and the WSDL Operations panel.

3 Click on the operation you want to map, expanding the corresponding panel.

Tables for input and output mapping appear.

NOTE

AR System sets default mappings automatically when you create a web service, or when you add an operation to a web service.

4 Select the Data Source type for your mapping.

a To have AR System automatically generate a basic XML schema, set the Data Source Type to Generated, click the Auto Map button, and skip to step f.

NOTE

You cannot use Developer Studio to modify XML elements in an external XML schema. To modify XML elements in an external schema, edit the XML schema file directly.

b If you are using an external XML schema, set the Data Source Type to XML Schema.

NOTE

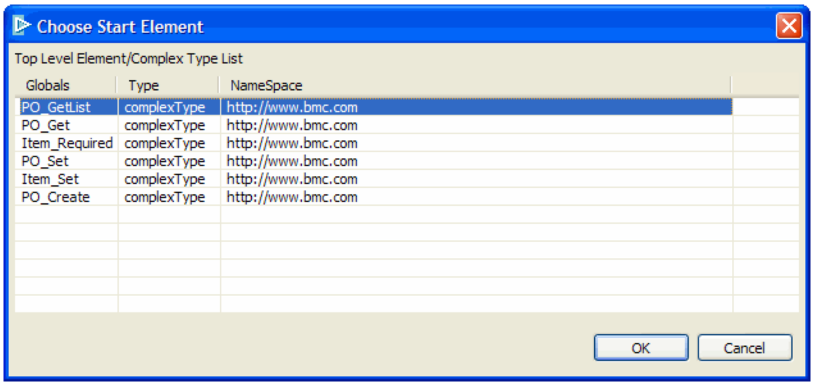
The XML Schema option is only available if you have specified a file name in the XML Schema panel, and the file has been loaded.

c Select Support XSI Type, if necessary. If this option is selected, the system specifies XSI Type for all XML data while creating XML documents.

d Click Choose, then click OK to verify that you want to change the mapping.

The Choose Start Element dialog box appears, displaying complex types and elements.

Figure B-2: Choose Start Element dialog box



Globals	Type	NameSpace
PO_GetList	complexType	http://www.bmc.com
PO_Get	complexType	http://www.bmc.com
Item_Required	complexType	http://www.bmc.com
PO_Set	complexType	http://www.bmc.com
Item_Set	complexType	http://www.bmc.com
PO_Create	complexType	http://www.bmc.com

e Select your start element or complex type. See "Importing an external XML schema" on page 358.

f Verify that your XML elements and complex types are displayed in the XML Data Type column, and the parent form and fields are displayed in the Form / Field column (see Figure B-4 on page 349).

TIP

To view the XML properties of an object in the XML Data Type column, hover your mouse over the object. To edit XML properties, select the object and click the ellipsis button [...].

5 Review the mappings, and make any necessary adjustments to forms, fields, and elements in your XML schema. (See Figure B-4 on page 349.)

To replace a form or field in the Form / Field column, select the object and click the adjacent ellipsis button [...].

6 Add child forms to the list of mapped objects.

a Right-click the ROOT element in the XML Data Type column.

b Choose New > Element > Form.

c Rename the new form element that appears under ROOT.

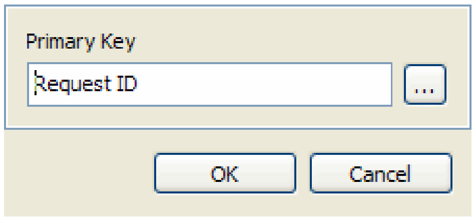
d Click the corresponding (blank) field in the Form / Field column.

e Click the ellipsis button [...], then select a form for mapping to your new form element.

7 In the row where your parent form is mapped, click the field in the Mapping Info column. The parent form is typically mapped to the ROOT element.

The Primary Key selector appears, as shown in Figure B-3.

Figure B-3: Primary Key selector


Primary Key: Request ID

8 Click the ellipsis button [...] and choose a field to use as the primary key.

This is the field that uniquely identifies the entries in the specified form. It is typically the Request ID field. The list of primary keys is limited to character fields specified as unique in Form > Form Properties > Indexes.

9 Click OK to close the Primary Key dialog box.

10 Map any other fields in the parent form to elements in the XML schema.

NOTE

When you create a filter to consume a web service, the output mapping table does not allow you to choose fields that only the system can modify.

You must map to the same data types. If you hover the cursor over the XML element name or field name, the tooltip shows the data type. See "Data types" on page 360.

11 In the row where one of your child forms is mapped, click the field in the Mapping Info column.

The Distinguishing Key and Foreign Key selectors appear. (These selectors are similar to the Primary Key selector in Figure B-3.)

12 Click the ellipsis button [...] and choose a value for Distinguishing Key. This is the value that uniquely identifies each of the detail items in the child form of any given parent entry.

NOTE

The AR System does not allow the mapping of an integer as the distinguishing key during the consumption of a web service.

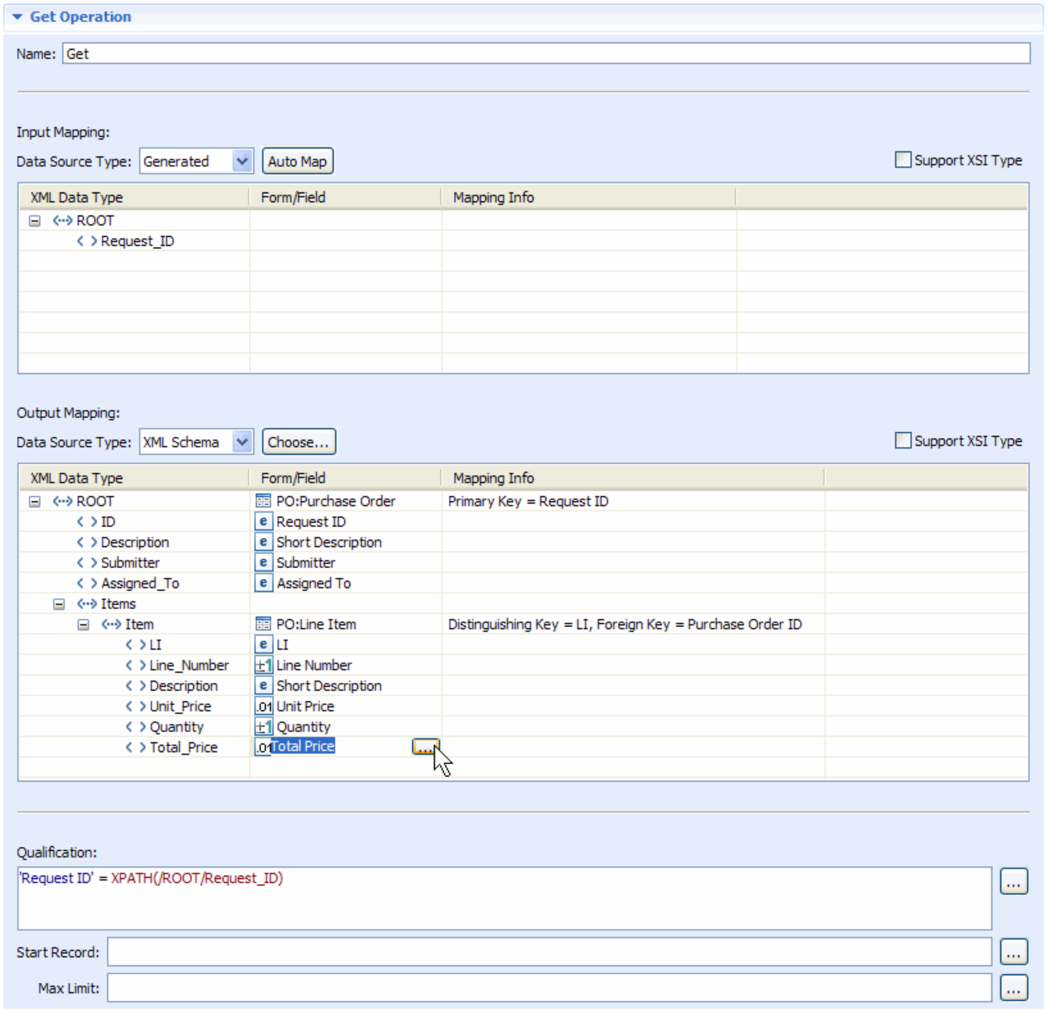
13 Click the ellipsis button [...] and choose a value for Foreign Key.

The combination of distinguishing key and foreign key should uniquely identify an entry in a child form. BMC Remedy Developer Studio does not enforce this relationship, but the AR System server returns an error if it detects noncompliance.

A field specified as foreign key should not be used in any field mapping, because this field is used by the system to store the foreign key. If it is mapped, the value of the mapped XML element is *usually* overridden by the foreign key value.

14 Map any other fields in the child form to XML elements.

Figure B-4: Mapping for a Get operation (complex document)



Name: Get

Input Mapping:

Data Source Type: Generated

XML Data Type	Form/Field	Mapping Info
ROOT		
Request_ID		

Output Mapping:

Data Source Type: XML Schema

XML Data Type	Form/Field	Mapping Info
ROOT	PO:Purchase Order	Primary Key = Request ID
ID	Request ID	
Description	Short Description	
Submitter	Submitter	
Assigned_To	Assigned To	
Items		
Item	PO:Line Item	Distinguishing Key = LI, Foreign Key = Purchase Order ID
LI	LI	
Line_Number	Line Number	
Description	Short Description	
Unit_Price	Unit Price	
Quantity	Quantity	
Total_Price	Total Price	

Qualification:

'Request ID' = XPATH(/ROOT/Request_ID)

Start Record:

Max Limit:

15 Repeat step 11 to step 14 to map other child forms to XML elements.

NOTE

The immediate children of all or choice element type cannot be mapped to another form.

To map choice nodes, map the choice node in the XML schema to the choice field in a form *before* mapping the choices.

16 Save your web service.

See "Supported schema constructs and AR System web service limitations" on page 101.

Using join forms in web services

You can use join forms in web services but only for parent forms; child forms cannot be join forms. For more information about join forms, see the *Form and Application Objects Guide*.

Primary keys

Join forms can be used for publishing or consuming in the same way as regular forms except for certain considerations when choosing a primary key in the Define form mapping dialog box. When you map the parent form, the primary key must be unique to the join form. The base forms which comprise the join form can each have a unique key. The Request ID of the join form is a concatenation of the Request IDs of the join base forms.

Because the primary key must be unique to the join form, this provides the following possibilities:

- For a `Create` operation:

  When you publish a web service using a join form, a `Create` operation is not automatically generated; you must make the `Create` operation yourself.

  If the join form is an inner join, the Primary key can be a Unique Index from any of the base forms that comprise the join form.

  If the join form is an outer join, the Primary key can be a Unique Index only from the primary base form.

  The primary key cannot be the Request ID field.
- For a `Set` operation:

  If the join form is an inner join, the primary key can be one of the following items:

  - Request ID of the join form
  - Request ID of any of the base forms
  - Unique Index from any of the base forms.

  If the join form is an outer join, the primary key can be one of the following items:

  - Request ID of the join form
  - Request ID of the primary base form
  - Unique Index of a field from the primary base form.

Output mapping

In a `Create` operation, if the web service base form is a join form, the output mapping is ignored and neither a document nor a Request ID is returned.

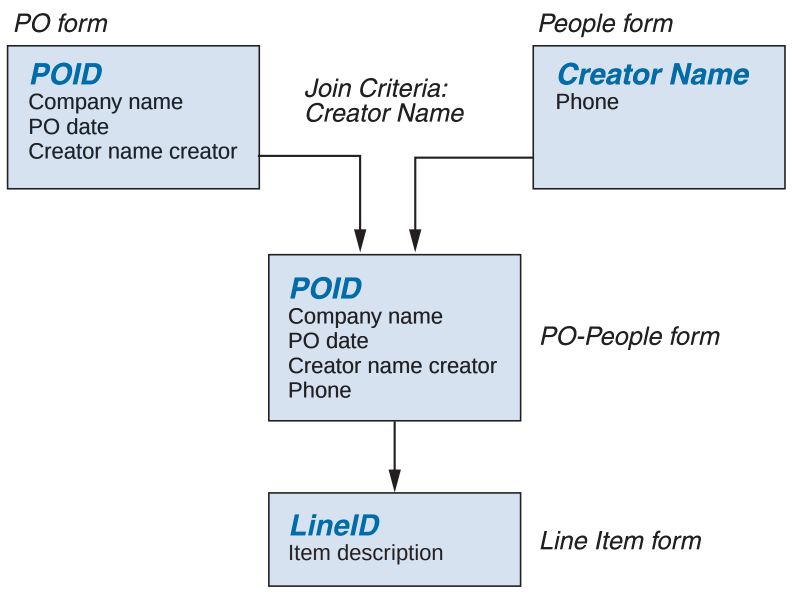
Example

In the following example, PO - People is an inner join form between the PO form and the People form.

The Unique key in the PO form is POID, the Unique key in the People form is Name, and the join criteria between the two forms are `Name (of PO form)= Name (of People form)`.

Because the Unique key POID is also unique in the join form, choose it as the Primary key.

Figure B-5: Join form in web services



XML editing

You can perform simple editing tasks with BMC Remedy Developer Studio or, for more complex documents, you can create and edit an XML schema outside AR System and import it. The following sections provide information about simple and complex XML editing.

Simple XML editing

BMC Remedy Developer Studio allows you to change the format of the XML code so that is compatible with your web service. The following changes are possible:

- You can add, delete, or rename the XML elements.
- You can create XML elements to enable grouping of existing XML elements. This grouping is purely cosmetic.
- You can use attributes instead of elements.
- You can set nillable, MinOccurs, and MaxOccurs attributes of elements.

▶ To add a new element or attribute

1 Right-click an element in the XML Data Type column to display a list of actions.

2 Choose New > Element or New > Attribute.

The new element or attribute appears one level below the selected item. You can add form elements, field elements, or field attributes.

3 Rename the new element or attribute.

▶ To change element ordering

1 Right-click an element in the XML Data Type column to display a list of actions.

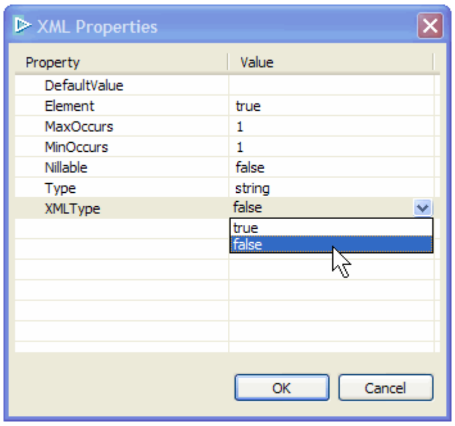
2 Click Move Up, Move Down, Move Left, or Move Right.

The element is moved, along with its mapping.

Object properties

The properties of an object indicate the information that the object provides. You can set various property values for the XML Schema objects as shown in the following figure.

Figure B-6: XML Properties dialog box



Property	Value
DefaultValue	
Element	true
MaxOccurs	1
MinOccurs	1
Nillable	false
Type	string
XMLType	false

To edit a property value, select the element in the XML Data Type column, then click the ellipsis button [...] to open the XML Properties dialog box.

- **DefaultValue—**This is the default value of an element unless it is overridden by the value in the actual document (incoming or outgoing).
- **Element**—If the value for Element is true, the object is an element. If the value is false, the object is an attribute. The ROOT element cannot be edited.
- **MaxOccurs—**This indicates maximum number of instances of the element. The default value is 1. A value of "unbounded" indicates that the element might be repeated any number of times. For example, a purchase order document might contain any number of line items.
- **MinOccurs—**This indicates number of instances of the element. The default value is 1. If the value is 0, the element might or might not be present.
- **Nillable—**Any leaf node (elements only) can be made nillable. Setting the nillable attribute to true or false can be done only in the mapping dialog boxes. The document is interpreted according to the rules described below.
- **Type—**This indicates the XML data type of the element.
- **XML Type—**Any leaf node (element or attribute) can be identified as XML Type. If XML Type is true, the data can be treated as XML string data. In this case, AR System does not treat it like a regular string (that is, the data is not encoded or decoded).

Handling null, empty, and missing values

There are many rules for handling null, empty, and missing values.

Elements and attributes mapped to fields

The rules can be divided into four groups.

- Incoming XML elements
- Incoming XML attributes
- Outgoing XML elements
- Outgoing XML attributes

AR System has two sources for incoming XML: as the request for an AR System published web service, or as a response from an external web service that AR System is consuming. Similarly there are two sources for outgoing XML: as the response from an AR System published web service, or the request to an external web service that AR System is consuming.

In these tables it is assumed that "name" is an XML element or attribute which is missing, empty, or nulled, and is mapped to an AR System field called "Name." The column headers are the design time properties—for example, "name" is defined with `minOccurs=0` and `nillable=false`. The row headers are run-time representations—for example, in the incoming XML packet "name" appears as `<name></name>`. The table specifies how AR System sets the XML element or attribute to or from the AR System field.

TIP

To render a null field, create an empty element with `xsi:nil=true` as an attribute. This is preferable to omitting the element in the request document, or creating an empty element with the nillable attribute set to false.

Incoming XML elements

	minOccurs=0 and nillable=false	minOccurs=0 and nillable=true	minOccurs=1 and nillable=false	minOccurs=1 and nillable=true
Missing ◂name▸	Name is not modified, or it is set to AR default.[1]	Name is not modified, or it is set to AR default.[1]	Invalid XML.[2]	Invalid XML.[2]
◂name▸◂/name▸ OR ◂name/▸	`Name=$NULL$` or xsd default [3]	`Name=$NULL$` or xsd default [3]	`Name=$NULL$` or xsd default [3]	`Name=$NULL$` or xsd default [3]
◂name xsi:nil="true▸◂/name▸ OR ◂name xsi:nil="true"/▸	Invalid XML.[5]	`Name=$NULL$` [4]	Invalid XML.[5]	`Name=$NULL$` [4]

[1] When an XML element is missing, AR System treats it the same way as a missing field. Therefore, in a create operation, the field to which the XML element is mapped assumes the AR System default value (or NULL if there is no default). In a set operation and in consumption, the field remains unchanged.
[2] When an XML element is missing, in spite of minOccurs=1, it is invalid XML. The client should not send such an XML packet, but if it does, AR System displays an error message.
[3] When the XML element has empty content, AR System first tries to use the xsd default if it exists. (There are two different defaults—the AR System default value and the xsd default value. For empty contents, AR System always uses the xsd default value.) Otherwise, it sets the field to NULL.
[4] When the XML element has xsi:nil=true, AR System sets the field to NULL and disregards the defaults.
[5] When the XML element has xsi:nil=true but is not defined with nillable=true, it is invalid XML. Clients should not send such an XML packet. Also, AR System sets this field to NULL, disregarding the defaults.

Incoming XML attributes

	use=optional	use=required
Missing ◂name▸	Name is set to xsd default, or it is not modified, or it is set to AR default.[1]	Invalid XML.[2]
name= ""	`Name=$NULL$` [3]	`Name=$NULL$` [3]

[1] If an attribute is defined with use=optional and the attribute is missing from the XML, AR System tries to use the xsd default. If the xsd default does not exist, AR System treats the attribute like a missing field. Therefore, in a create operation, the field to which this attribute is mapped assumes the AR System default value (or NULL if there is no default). In a set operation and in consumption, the field remains unchanged.
[2] If an attribute is defined with use=required, it should not be missing. Otherwise, the XML is invalid and clients should not send such an XML packet. AR System displays an error message.
[3] If an attribute has an empty value, AR System sets the mapped field to NULL and disregards the defaults.

Outgoing XML elements

	minOccurs=0 and nillable=false	minOccurs=0 and nillable=true	minOccurs=1 and nillable=false	minOccurs=1 and nillable=true
Name is $NULL$	Missing name [2]	`<name xsi:nil=“true”/>` [1]	`<name/>` [3]	`<name xsi:nil=“true”/>` [1]
Name is “”	`<name/>`	`<name/>` [4]	`<name/>` [4]	`<name/>` [4]
◂name▸ is not mapped	Missing name	Missing name	Invalid XML.	Invalid XML.

[1] If a field is null, AR System generates the XML as `xsi:nil=true`. However, it can do so only if `nillable=true`.
[2] If nillable is false, AR System doesn't generate the element at all for null fields. However, it can do so only if `minOccurs=0`.
[3] If nillable is false and `minOccurs=1`, AR System generates an element with empty content.
[4] If a character field contains an empty string, AR System generates an element with empty content. AR System fields with empty strings are extremely unusual; they can be specified only with the driver program or an API call.

Outgoing XML attributes

	use=optional	use=required
Name is $NULL$	`name=""` [1]	`name=""` [1]
Name is ""	`name=""` [2]	`name=""` [2]
◂name▸ is not mapped	Missing name	Invalid XML.

[1] If a field is null, AR System generates an attribute with empty content.
[2] If a character field contains an empty string, AR System generates an attribute with empty content. AR System fields with empty strings are extremely unusual; they can be specified only with the driver program or an API call.

Elements mapped to forms

While elements mapped to fields can only have `maxOccurs=1`, elements mapped to forms can have `maxOccurs>1`. (Actually, elements mapped to fields can have `maxOccurs>1`, but at run time at most one element should appear in the incoming XML.)

Incoming XML elements

For incoming XML, the base form can be mapped only to an element with `maxOccurs=1`. (It is acceptable if `maxOccurs>1` at design time, but at run time there is at most one element.)

The child forms can be mapped to elements with `maxOccurs>1`. If the number of XML elements does not fall in the range set by minOccurs and maxOccurs, it is invalid XML and the client should not send a document containing such XML. However AR System ignores the minOccurs, maxOccurs constraints while parsing this XML.

Outgoing XML elements

For outgoing XML, the base form can be mapped to an element with `maxOccurs>1` in case of publishing and an operation of type get. This implies that multiple entries in the base form are to be retrieved. If the number of entries in the base form is less than the minOccurs, AR System returns an error. If the number of entries is more than the maxOccurs, AR System returns only until the maxOccurs amount.

Child forms can be mapped to elements with `maxOccurs>1`. If the number of matching entries in the child form does not fall in the range set by minOccurs and maxOccurs, AR System returns an error.

Flat mapping

The typical procedure is to create the XML elements and then map them to the fields. However, if you are creating a flat mapping, you can combine these two steps into one. Remove the fields that you do not want to map and then click the Generate Schema button on the Mapping dialog box. This creates XML elements that are automatically mapped.

▶ To remove an existing field from the mapping list

1 In the Form / Field column, highlight the field name you want to remove.

2 Use the Backspace or Delete key to remove the field name.

The corresponding XML element is not removed.

▶ To remove a mapping

1 In the mapping table, right-click the mapping you want to remove.

Each mapping is represented by a row in the table.

2 Click Remove Selected.

XML editing is allowed only while you publish a web service from AR System. When you consume an external web service, the XML format is decided by the external web service and you must conform to it. BMC Remedy Developer Studio disables all editing features in that case. You can choose only which fields to map to which XML elements.

XML documents can specify the data type in the document itself instead of in the XML schema. If you want the output document to contain XSI Type information so that the consumers of the web service can process the document correctly, check the Support XSI Type option above the mapping table. In this case, the system generates XSI Type information.

Advanced XML editing

The mapping tables in BMC Remedy Developer Studio allow for simple XML editing. For more complex editing, you can use an advanced XML schema editing tool, such as XMLSpy. XML editing tools are not included with AR System. If you have a thorough understanding of XML schemas, you can write them in a simple text editor.

An online tutorial on XML schemas is available at this website:

`http://www.xfront.com/index.html#schema`

After you design an XML schema, you can import it into BMC Remedy Developer Studio. After you import it, however, the XML editing features in BMC Remedy Developer Studio are disabled. If you want to use BMC Remedy Developer Studio to edit your XML schema, you must create the schema within Developer Studio.

If you use an old XML schema editor, the namespace for the XML schema might be 2000 or 1999. BMC supports only namespaces of 2001—that is, with the declaration `http://www.w3.org/2001/XMLSchema`. Using an XSD file with an older namespace might produce unexpected mapping results.

Importing an external XML schema

One external XML schema can be used for all the mappings of one web service. However, this XML schema can include or import other XML schemas, so to use multiple XML schemas, create an XML schema that includes or imports all the schemas you want to use.

▶ To import an external XML schema

1 Expand the XML Schema panel in Developer Studio.

2 Specify your XML Schema (XSD file) in the XML Schema field, or use the ellipsis button [...] to browse to the XSD file.

If your schema definition is spread over multiple XSD files interlinked together using import or include, you must type the URL of the topmost XSD file—that is, the one that is not included or imported by others.

You can also enter a file system path to your XSD file, but the URL to the XSD file is referenced in your generated WSDL file, so your path must be accessible over the network. You should create a web server directory to hold your XSD files, and enter the http path to this directory in the XML Schema field.

3 Click the Reload button.

4 Choose the XML Schema Source type in the drop-down list.

a If you entered a local file system path for your XSD, select Embedded. In this case, AR System loads the specified XML schema and all dependent XML schemas. It stores the entire XSD file and all other files that the XSD file includes or imports. The WSDL also has all the XSDs embedded in the types section. This is the default option.

b If you entered a network accessible path (http or ftp) for the XSD, you can select Linked. In this case, AR System does not store the XSD files, but it does store a reference to the specified schema in the web service object and in the WSDL file. Some WSDL parsing tools (early versions of Microsoft.NET and MSSOAP) do not support these kind of WSDLs.

A system-generated schema is always embedded in the WSDL.

If your schema definition is spread over multiple XSD files linked together using import or include, you must type the URL of the topmost XSD file—that is, the one which is not included or imported by anyone else.

5 In the WSDL Operations panel, expand an operation.

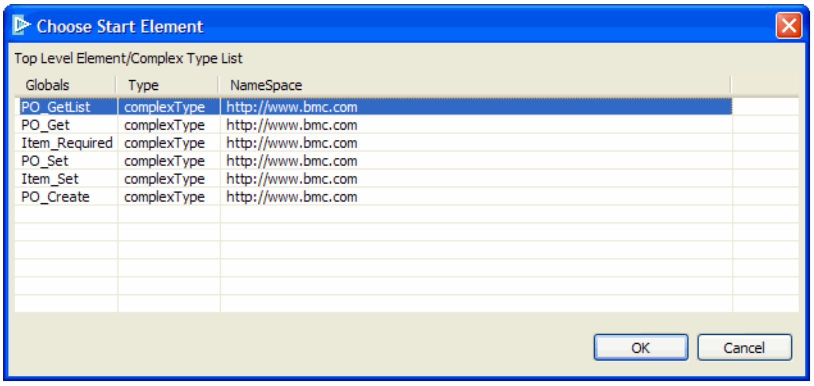
6 In the Data Source Type list, select XML Schema, and click the Choose button.

7 Click OK to verify that you want to change the mapping.

8 Select a start element.

Separate mappings can be based on separate global elements, but all of them must come from the same XML schema. BMC Remedy Developer Studio displays a list of global elements and global complex types that either reside in your XSD file, or are included or imported into the file.

Figure B-7: Choose start element dialog box



Globals	Type	NameSpace
PO_GetList	complexType	http://www.bmc.com
PO_Get	complexType	http://www.bmc.com
Item_Required	complexType	http://www.bmc.com
PO_Set	complexType	http://www.bmc.com
Item_Set	complexType	http://www.bmc.com
PO_Create	complexType	http://www.bmc.com

If you select an element, the element and all its successors are added as a child of the ROOT element. If you select a complexType, the contents of the complexType is added as a child of the ROOT.

If you specify and load a different schema, AR System verifies that global elements or complex types referred to in the current mappings are compatible with the new schema and informs you of incompatible types. If you agree to update the existing mappings, AR System updates them for you. If there are no overlapping global or complex types, AR System preserves the content of the original mapping.

BMC Remedy Developer Studio does not support all features of XML schemas, and when using a web service there are restrictions that apply to the external WSDL file. For more information about limitations, see the AR System *Release Notes*.

Data types

When mapping an XML element to an AR System field, BMC Remedy Developer Studio allows only compatible data types, both for consuming and publishing.

AR System data types	XML schema data types
`Character`	`string, duration, anyURI, QName, NOTATION, normalizedString, token, language, NMTOKEN, Name, NCName, ID, IDREF, ENTITY, integer, positiveInteger, nonPositiveInteger, negativeInteger, nonNegativeInteger`
`Status History`	`string`
`Diary`	`string`
`Date/Time`	`dateTime`

`Date`	`date, gYearMonth, gYear, gMonthDay, gDay, gMonth`
	Defaults:
	`YEAR - 1000 (leap year), month - 01, day - 01`

`Time`	`time`
`Currency (value)`	`decimal`
`Currency (code)`	`string`
`Currency (conversion date)`	`dateTime`
`Integer`	`int, long, unsignedLong, unsignedInt, boolean, short, byte, unsignedShort, unsignedByte`
`Real`	`double, float`
`Decimal`	`decimal`
`Drop-Down, Radio, Check Box`	`string`
`Attachment (name)`	`string`
`Attachment (data)`	`base64Binary`
`Attachment (original size)`	`int, long, unsignedLong, unsignedInt, boolean, short, byte, unsignedShort, unsignedByte`

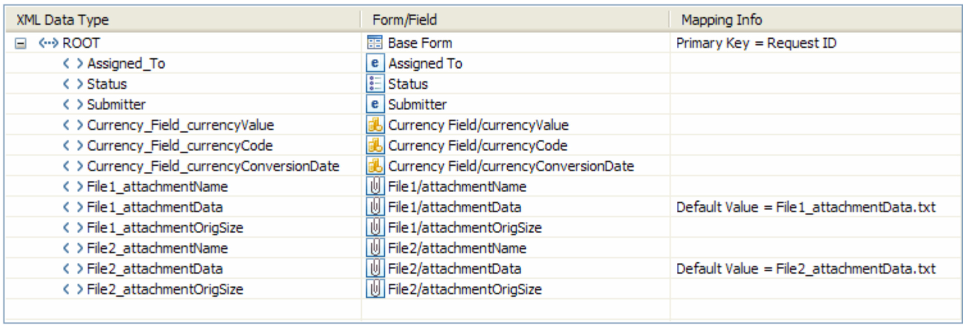
NOTE

AR System web services do not support list and union data types. AR System converts list data types `IDREFS`, `ENTITIES`, and `NMTOKENS` to strings.

The following complex AR System fields are treated as exceptions:

- When you retrieve a diary field from AR System, the diary field is treated as a long character field containing all the historical diary entries separated by a special separator character. When you send a diary field to AR System, you send only the current entry.
- The Status History field is treated similarly to a diary field, but you cannot send a status history entry to AR System.
- Each currency and attachment field consists of three parts, and each part needs to be mapped separately (see Figure B-8).

Figure B-8: Mapping currency and attachment fields



XML Data Type	Form/Field	Mapping Info
ROOT	Base Form	Primary Key = Request ID
Assigned_To	Assigned To	
Status	Status	
Submitter	Submitter	
Currency_Field_currencyValue	Currency Field/currencyValue	
Currency_Field_currencyCode	Currency Field/currencyCode	
Currency_Field_currencyConversionDate	Currency Field/currencyConversionDate	
File1_attachmentName	File1/attachmentName	
File1_attachmentData	File1/attachmentData	Default Value = File1_attachmentData.txt
File1_attachmentOrigSize	File1/attachmentOrigSize	
File2_attachmentName	File2/attachmentName	
File2_attachmentData	File2/attachmentData	Default Value = File2_attachmentData.txt
File2_attachmentOrigSize	File2/attachmentOrigSize	

Appendix

C ARDBC LDAP example: Accessing inetorgperson data

This appendix contains an example of how to create a vendor form associated with a collection of objects (using the `inetorgperson` object class) in an LDAP directory service and how to attach data to it.

The following topics are provided:

- Creating the inetorgperson vendor form (page 364)
- Attaching fields to represent inetorgperson data (page 365)
- Defining a filter to generate a DN (page 367)

NOTE

You must install and configure your ARDBC plug-in before you can create a vendor to use the plug-in. See "Creating C plug-ins" on page 110 and "Configuring the ARDBC LDAP plug-in" on page 137.

Creating the inetorgperson vendor form

The `inetorgperson` object class is often present by default on some LDAP directory services, such as iPlanet and OpenLDAP. To use the example in this appendix if your LDAP service does not contain the `inetorgperson` object class, replace the `objectclass` filter in the `inetorgperson` vendor form. The data that corresponds to your new object class should contain the following attributes: `uid`, `sn`, `dn cn`, `ou`, and `objectclass`. Instructions about how and when to change the `objectclass` file are presented later in this section. The form and workflow are provided with the LDAP plug-in software distribution in the `inetorgperson.def` file, typically found in the *`ARSystemServerInstallDir`*`\Plugins\ARDBC` folder.

NOTE

The introgperson vendor form that is installed by default, is a sample vendor form with default vendor information. The vendor information needs to be configured for a specific environment before the form can be used. The default data is just a placeholder and will not work.

▶ To create a vendor form using the inetorgperson objectclass

1 Start BMC Remedy Developer Studio and log in to an AR System server.

2 From the AR System Navigator list on the left side of the screen expand All Objects and then select Forms.

3 From the forms list on the right side of the screen, select the default sample form, inetorgperson of type Vendor.

The Vendor Form and data are displayed.

4 Adjust any of the fields as needed to configure the vendor information for your implementation.

NOTE

The corresponding URL for `inetorgperson` is:

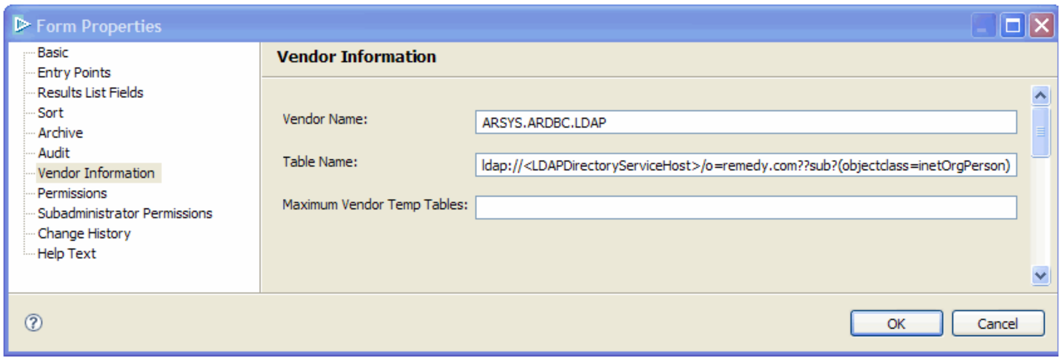
```
ldap://LDAPDirectoryServiceHost/o=remedy.com??
sub?(objectclass=inetOrgPerson)
```

If your LDAP server does not contain the `inetorgperson` objectclass, select a comparable objectclass, such as `person`.

5 Click File > Save to save the vendor form.

You can modify the LDAP search URL at any time. In the Form Properties dialog box, you can also configure the form to use the ARDBC LDAP plug-in. Other ARDBC plug-ins require that you enter a different plug-in name and might not use an LDAP search URL format to define a collection of objects.

Figure C-1: Form Properties window, showing LDAP search URL



Vendor Information

Vendor Name: ARSYS.ARDBC.LDAP

Table Name: ldap://<LDAPDirectoryServiceHost>/o=remedy.com??sub?(objectclass=inetOrgPerson)

Maximum Vendor Temp Tables:

NOTE

You might need to further refine the base-distinguished name portion of the URL in the Table Name parameter of the form properties to refine the search further. Some LDAP servers do not return the table of results if the base-distinguished name used to search for entries is not specific enough.

6 To add or delete fields from a vendor form:

- To add a field to the vendor form, choose Form > Add Fields from *`tableName`*. Select a field to add from the Add Fields dialog box, and click OK.
- To delete a field from the vendor form, click a field and choose Edit > Delete. Deleted fields are returned to the Add Fields list for later access, if needed. This only deletes the AR System field. It does not remove the column from the database table.

Attaching fields to represent inetorgperson data

After you create a vendor form, you populate the form with fields that contain data from your `inetorgperson` data source. This section describes how to add a field to represent the User ID in the `inetorgperson` example.

▶ To add a field to represent the uid (User ID) attribute in the inetorgperson example

1 In BMC Remedy Developer Studio, open the vendor form you created earlier.

2 Right-click in the form and choose Add Fields from *`tableName`*.

The *`tableName`* variable is the object represented by `ldap://`*`LDAPDirectoryServiceHost`*`/o=remedy.com??sub?(objectclass=inetorgperson)`.

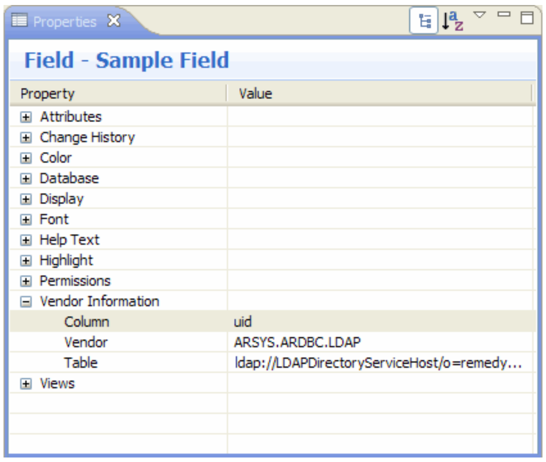
3 In the Add Fields dialog box, select a field to add and click OK.

4 Position the field on your form, as required.

5 Select the field to display its properties in the Properties tab.

6 In the Properties tab, expand the Vendor Information category (see Figure C-2).

Figure C-2: Field Properties tab—vendor information



Field - Sample Field

Property	Value
⊞ Attributes	
⊞ Change History	
⊞ Color	
⊞ Database	
⊞ Display	
⊞ Font	
⊞ Help Text	
⊞ Highlight	
⊞ Permissions	
⊟ Vendor Information	
Column	uid
Vendor	ARSYS.ARDBC.LDAP
Table	ldap://LDAPDirectoryServiceHost/o=remedy...
⊞ Views	

7 Change the value of the Column property to `uid`.

8 Click File > Save.

▶ Alternative method of adding a field to represent the uid (User ID) attribute

1 Open the form where you want to add a field.

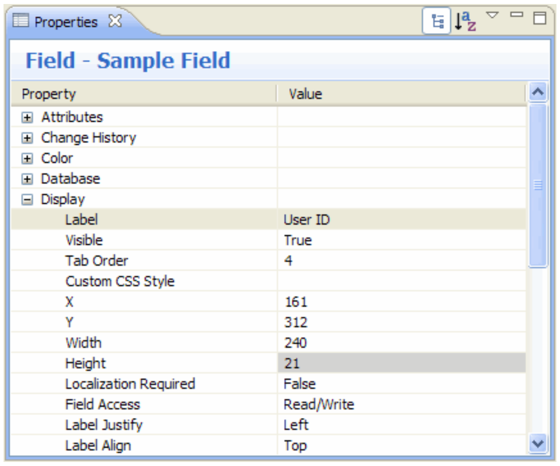
2 Add a character field to the form by choosing Form > Create a New Field > Character.

3 Select the field to display its properties in the Properties tab.

4 In the Properties tab, expand the Display category (see Figure C-3).

5 Change the value of the Label property to `User ID`.

Figure C-3: Field Properties tab—display information



Field - Sample Field

Property	Value
⊞ Attributes	
⊞ Change History	
⊞ Color	
⊞ Database	
⊟ Display	
Label	User ID
Visible	True
Tab Order	4
Custom CSS Style	
X	161
Y	312
Width	240
Height	21
Localization Required	False
Field Access	Read/Write
Label Justify	Left
Label Align	Top

6 In the Properties tab, expand the Database category.

7 Change the value of the Entry Mode property to `Required`.

8 In the Properties tab, expand the Vendor Information category.

9 Change the value of the Column property to `uid`.

10 Position the field on your form, as required.

11 Click File > Save.

Defining a filter to generate a DN

In the `inetorgperson` example, an object's distinguished name looks something like this:

```
uid=abarnes, ou=People, o=remedy.com
```

The following procedure shows how to create an AR System filter to assemble the distinguished name using the `inetorgperson` example.

▶ To define an AR System filter to construct the distinguished name using the inetorgperson example

1 In BMC Remedy Developer Studio, choose File > New > Filter.

2 Select the server where you want to create the filter, and click Finish.

3 Right-click the Associated Forms panel, then choose Expand All Panels.

4 In the Associated Forms panel, click the Add button.

5 In the Form Selector dialog box, select the `inetorgperson` form and click OK.

6 In the Execution Options panel, select the Submit check box.

7 Right-click the If Actions panel, then choose Add Action > Set Fields.

A Set Fields subpanel appears, which includes a field-value table (see Figure C-4).

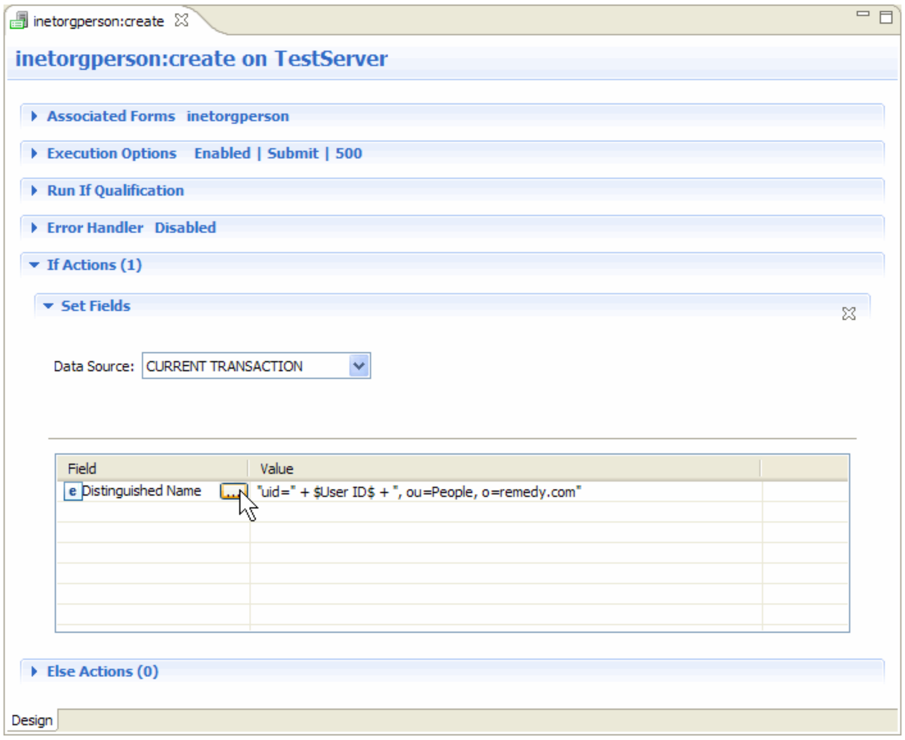
8 Click the first cell in the Field column, then click the ellipsis button [...].

9 Use the Field Selector to choose the Distinguished Name field, then click OK.

10 In the corresponding Value cell, type the following expression:

```
"uid=" + $User ID$ +  ", ou=People, o=remedy.com"
```

Figure C-4: Creating the inetorgperson filter


inetorgperson:create

inetorgperson:create on TestServer

Associated Forms inetorgperson

Execution Options Enabled | Submit | 500

Run If Qualification

Error Handler Disabled

If Actions (1)

Set Fields

Data Source: CURRENT TRANSACTION

Field	Value
Distinguished Name	"uid=" + $User ID$ + ", ou=People, o=remedy.com"

Else Actions (0)

Design

11 Click File > Save.

12 Name your filter `inetorgperson:create`, then click OK.

You can now log in to BMC Remedy User and open the `inetorgperson` task to search and create entries.

Summary of fields

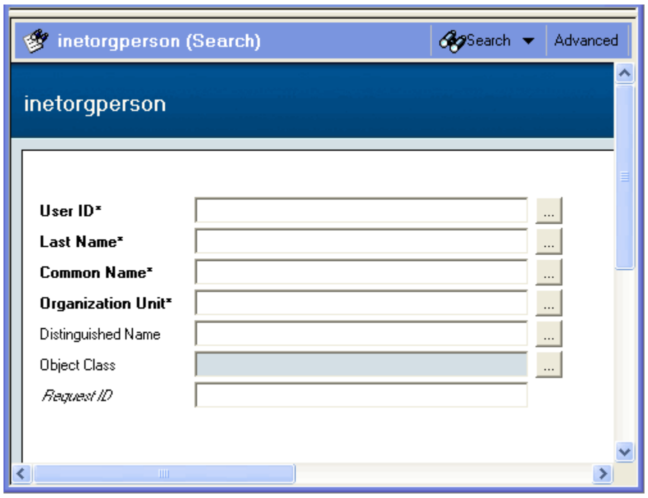
In the `inetorgperson` example, the following fields are needed:

Field	Field properties
`Distinguished Name`	Entry Mode: Optional Read / Write Default Value: none Vendor Information Column: `dn`
`Object Class`	Entry Mode: Required Read Only Default Value: top, person, organizationalPerson, inetorgperson Vendor Information Column: `objectclass[*,]`

Field	Field properties
`Last Name`	Entry Mode: Required Read / Write Default Value: none Vendor Information Column: `sn`
`Common Name`	Entry Mode: Required Read / Write Default Value: none Vendor Information Column: `cn`
`Organization Unit`	Entry Mode: Required Read / Write Default Value: People Vendor Information Column: `ou[*,]`

In this example, the form looks like the form in Figure C-5.

Figure C-5: The inetorgperson form in BMC Remedy User



User ID*	
Last Name*	
Common Name*	
Organization Unit*	
Distinguished Name	
Object Class	
Request ID	

Appendix

D Web service examples

This section provides examples of creating and consuming web services of various complexity.

The following topics are provided:

- Example 1: Publishing a simple flat document (page 372)
- Example 2: Consuming a simple flat document (page 377)
- Example 3: Publishing a complex document (page 381)
- Example 4: Consuming a complex document (page 394)

For more information, see Chapter 6, “Web services.”

Example 1: Publishing a simple flat document

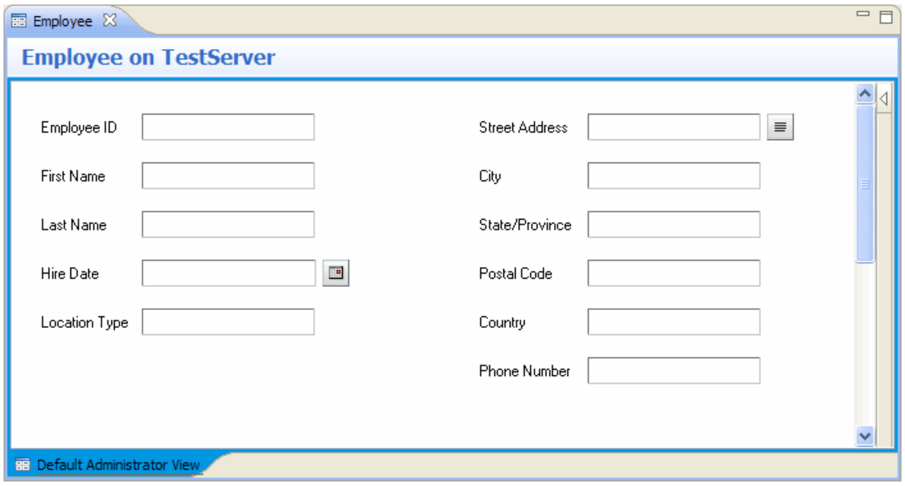
In this example, you publish a web service that includes default operations of `Create`, `Get`, `GetList`, `Set`, and `Service` for an employee record.

▶ To publish a simple flat document

1 Create a form that displays your employee data.

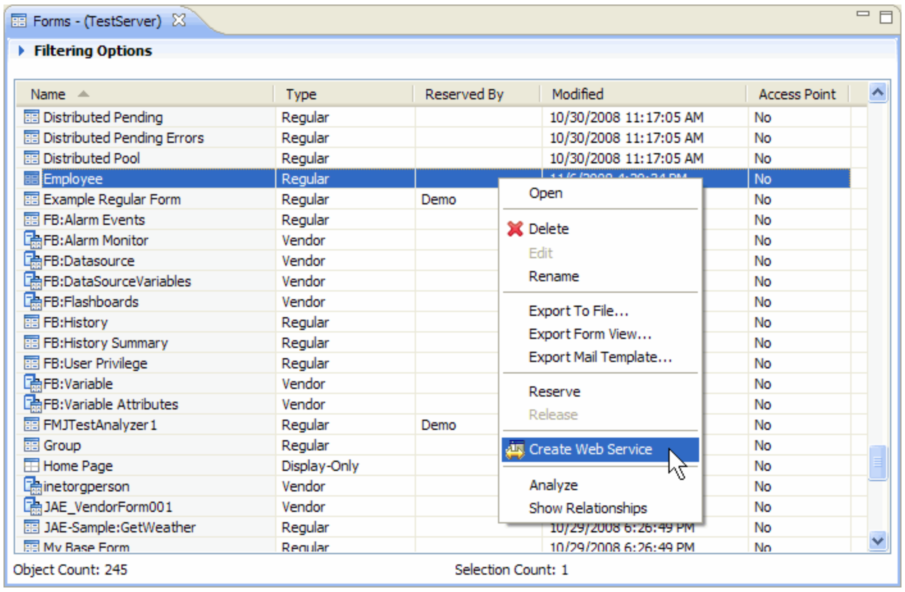
This is the base form used to create your web service (see "Forms and field mappings for web services" on page 64). Figure D-1 shows a sample form called Employee. All the fields are character fields.

Figure D-1: Sample form for employee record



2 On the Forms tab, right-click the form name to display its shortcut menu (see Figure D-2).

Figure D-2: Forms tab showing list of forms and web services selection



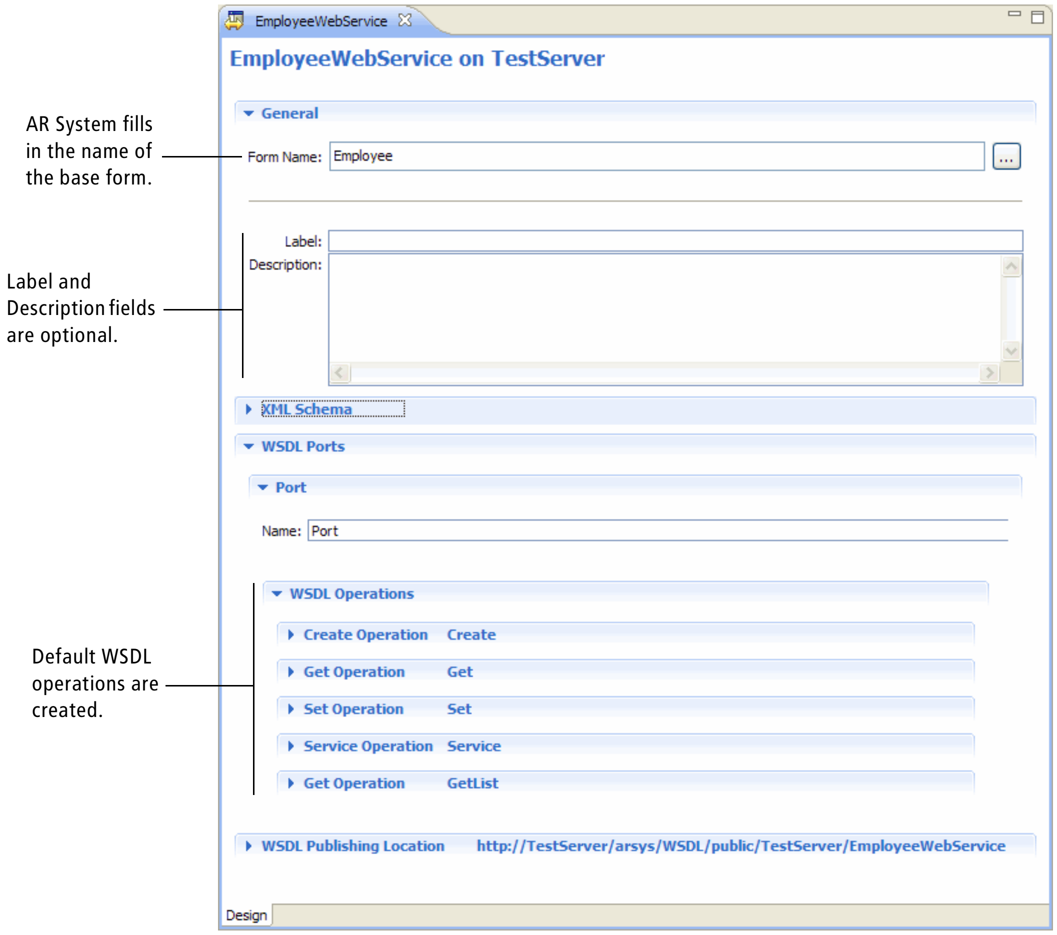
Name	Type	Reserved By	Modified	Access Point
Distributed Pending	Regular		10/30/2008 11:17:05 AM	No
Distributed Pending Errors	Regular		10/30/2008 11:17:05 AM	No
Distributed Pool	Regular		10/30/2008 11:17:05 AM	No
Employee	Regular			No
Example Regular Form	Regular	Demo		No
FB:Alarm Events	Regular			No
FB:Alarm Monitor	Vendor			No
FB:Datasource	Vendor			No
FB:DataSourceVariables	Vendor			No
FB:Flashboards	Vendor			No
FB:History	Regular			No
FB:History Summary	Regular			No
FB:User Privilege	Regular			No
FB:Variable	Vendor			No
FB:Variable Attributes	Vendor			No
FMJTestAnalyzer1	Regular	Demo		No
Group	Regular			No
Home Page	Display-Only			No
inetorgperson	Vendor			No
JAE_VendorForm001	Vendor			No
JAE-Sample:GetWeather	Regular		10/29/2008 6:26:49 PM	No
My Base Form	Regular		10/29/2008 6:26:49 PM	No

Object Count: 245 Selection Count: 1

3 Select Create Web Service from the menu.

A new Web Service tab appears, with default settings based on your form.

Figure D-3: EmployeeWebService web service


AR System fills in the name of the base form.
Label and Description fields are optional.
Default WSDL operations are created.
EmployeeWebService
Design

EmployeeWebService on TestServer

General

Form Name: Employee

Label:

Description:

XML Schema

WSDL Ports

Port

Name: Port

WSDL Operations

Create Operation	Create
Get Operation	Get
Set Operation	Set
Service Operation	Service
Get Operation	GetList

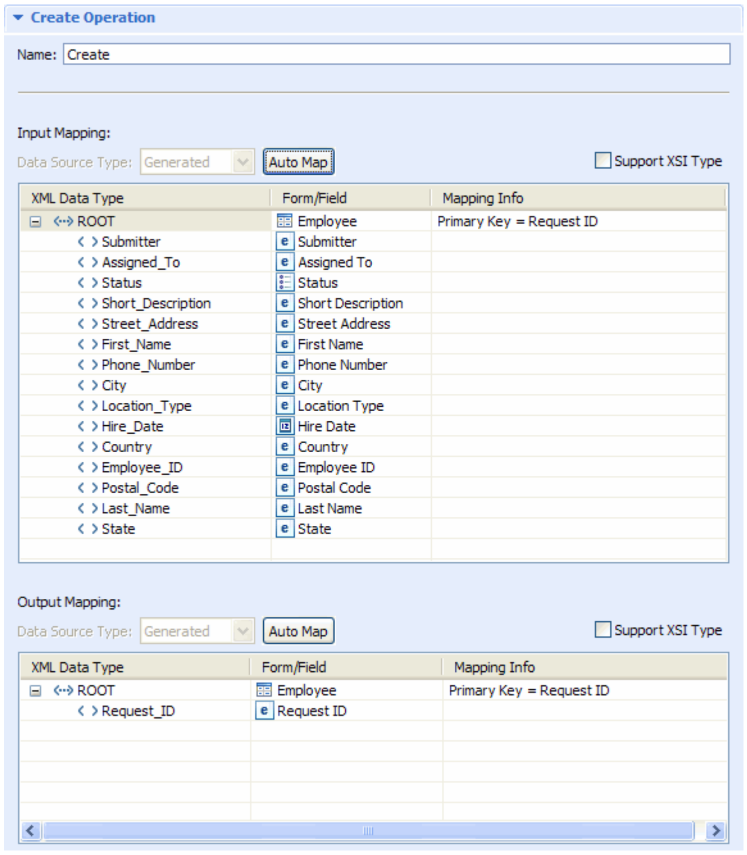
WSDL Publishing Location http://TestServer/arsys/WSDL/public/TestServer/EmployeeWebService

BMC Remedy Developer Studio automatically fills in the name of the base form (Employee). The Label and Description fields are optional. Default operations are automatically displayed on the WSDL Operations panel.

4 Select an operation from the WSDL Operations list, and expand the corresponding panel.

The fields on your Employee form are mapped to XML-compliant element names in a default XML schema, as shown in Figure D-4. (You do not need to modify the default mappings for this example.)

Figure D-4: EmployeeWebService—Mapping tables for a Create operation



Create Operation

Name: Create

Input Mapping:

Data Source Type: Generated

XML Data Type	Form/Field	Mapping Info
ROOT	Employee	Primary Key = Request ID
Submitter	Submitter	
Assigned_To	Assigned To	
Status	Status	
Short_Description	Short Description	
Street_Address	Street Address	
First_Name	First Name	
Phone_Number	Phone Number	
City	City	
Location_Type	Location Type	
Hire_Date	Hire Date	
Country	Country	
Employee_ID	Employee ID	
Postal_Code	Postal Code	
Last_Name	Last Name	
State	State	

Output Mapping:

Data Source Type: Generated

XML Data Type	Form/Field	Mapping Info
ROOT	Employee	Primary Key = Request ID
Request_ID	Request ID	

5 Close the WSDL Operations panel.

6 Set the permissions to Public.

The Permissions property is modified from the Properties tab corresponding to EmployeeWebService. This step is important if you publish your web service over an internet or intranet for general use.

7 Choose File > Save, and name your web service EmployeeWebService.

8 Click the WSDL Publishing Location tab (see Figure D-5 on page 376).

A sample URL for your WSDL file is displayed in the Specify mid-tier's WSDL handler URL field.

9 Modify the URL appropriately for your configuration:

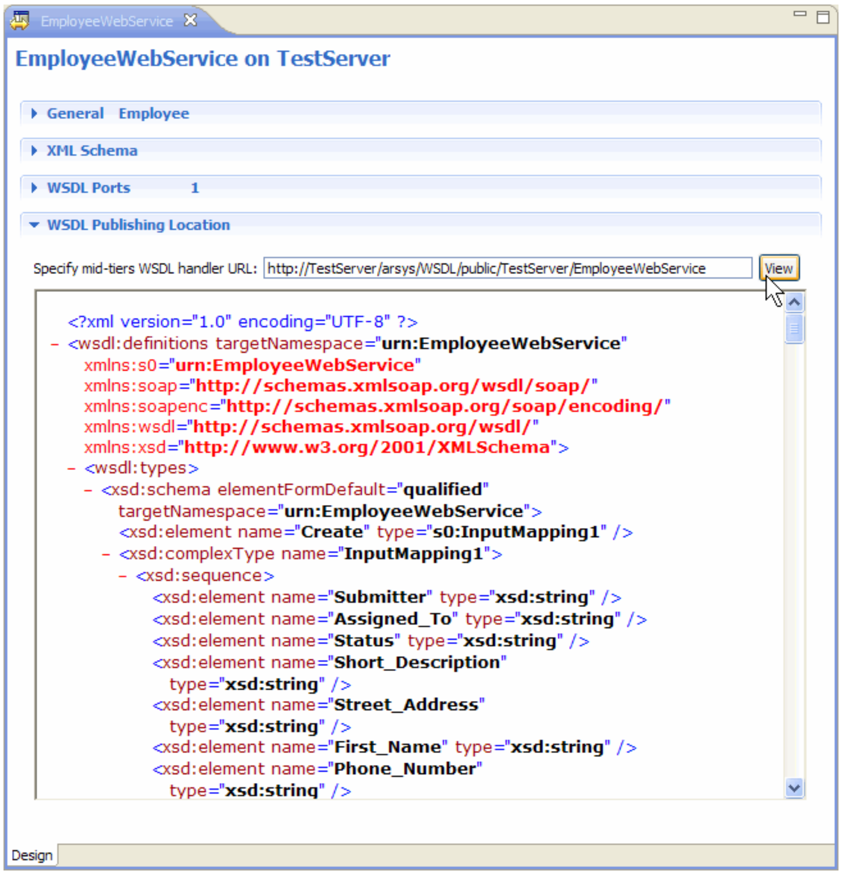
a Replace `<midtier_server>` with the name of the web server where the mid tier is running.

b Add `/public` after WSDL.

For example, if the mid tier server is `TestServer`, you would use this URL:

```
http://TestServer/arsys/WSDL/public/TestServer/EmployeeWebService
```

Figure D-5: EmployeeWebService—WSDL tab on Create Web Service window


EmployeeWebService
EmployeeWebService on TestServer
General Employee
XML Schema
WSDL Ports 1
WSDL Publishing Location
Specify mid-tiers WSDL handler URL: http://TestServer/arsys/WSDL/public/TestServer/EmployeeWebService
View
Design

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="urn:EmployeeWebService"
    xmlns:s0="urn:EmployeeWebService"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <xsd:schema elementFormDefault="qualified"
        targetNamespace="urn:EmployeeWebService">
        <xsd:element name="Create" type="s0:InputMapping1" />
      - <xsd:complexType name="InputMapping1">
        - <xsd:sequence>
            <xsd:element name="Submitter" type="xsd:string" />
            <xsd:element name="Assigned_To" type="xsd:string" />
            <xsd:element name="Status" type="xsd:string" />
            <xsd:element name="Short_Description"
              type="xsd:string" />
            <xsd:element name="Street_Address"
              type="xsd:string" />
            <xsd:element name="First_Name" type="xsd:string" />
            <xsd:element name="Phone_Number"
              type="xsd:string" />
```

10 Click File > Save to save your changes to EmployeeWebService.

Administrators with the appropriate SOAP protocol can now access this WSDL file with any browser.

11 (optional) Click View to display the contents of your WSDL file.

12 (optional) If prompted, enter a user name and password and click Login.

Example 2: Consuming a simple flat document

In this example, you access an external web service and, using a Set Fields filter action, set the data into an AR System form. The external web service takes a stock ticker symbol as input and returns the 20-minute delayed stock quotation for that stock.

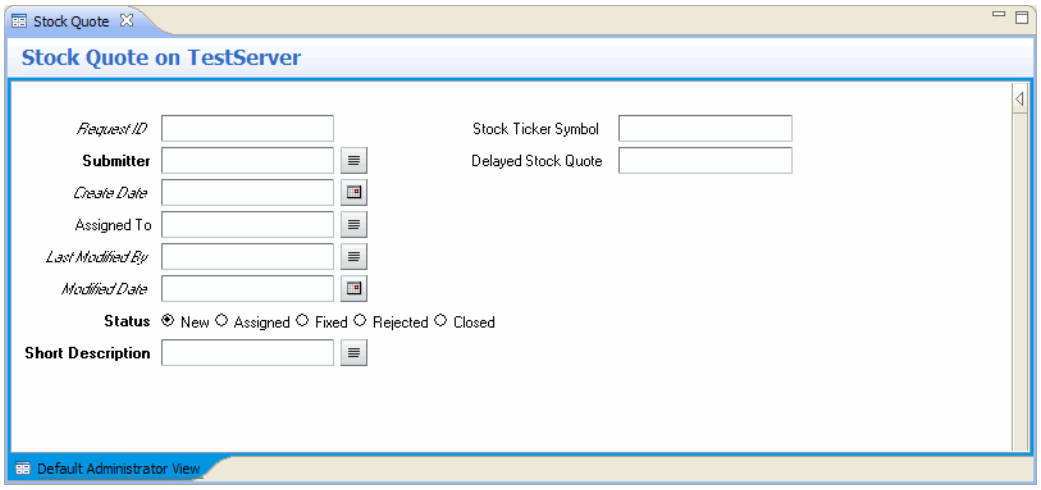
▶ To consume a simple flat document

1 In BMC Remedy Developer Studio, create a form called Stock Quote.

2 Add two fields:

- A character field called Stock Ticker Symbol in which you enter the stock ticker symbol (for example, BMC)
- A decimal field called Delayed Stock Quote, in which the stock quotation is set when information is received from the external web service

Figure D-6: Stock Quote form in BMC Remedy Developer Studio



3 In BMC Remedy Developer Studio, choose File > New > Filter.

4 Select the server on which you want to create the filter, and click Finish.

5 Right-click the Associated Forms panel, then choose Expand All Panels.

6 In the Associated Forms panel, click the Add button (see Figure D-7).

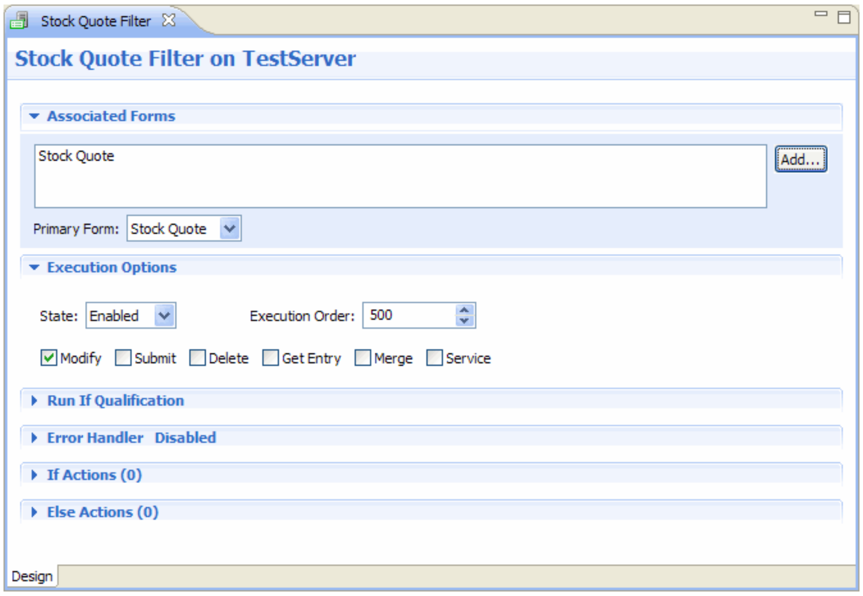
7 In the Form Selector dialog box, select the Stock Quote form and click OK.

8 From the State list, choose Enabled.

9 In the Execution Order field, enter the execution order for the filter.

10 In the Execution Options panel, select Modify.

Figure D-7: Stock Quote filter—Associated Forms and Execution Options panels


Stock Quote Filter

Stock Quote Filter on TestServer

Associated Forms

Stock Quote

Primary Form: Stock Quote

Execution Options

State: Enabled   Execution Order: 500

[✓ Modify] Submit Delete Get Entry Merge Service

Run If Qualification

Error Handler Disabled

If Actions (0)

Else Actions (0)

11 Right-click the If Actions panel, choose Add Action, and select Set Fields.

A new panel appears, where you configure the Set Fields action (see Figure D-8.)

12 From the Data Source list, choose WEB SERVICE.

13 From the Server Name list, select the server on which to store the web service mappings as a server object.

14 In the WSDL File field, enter the URL for the WSDL file of the external web service.

`http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl`

NOTE

This WSDL reference might not work if its website is not functioning.

15 Click Reload.

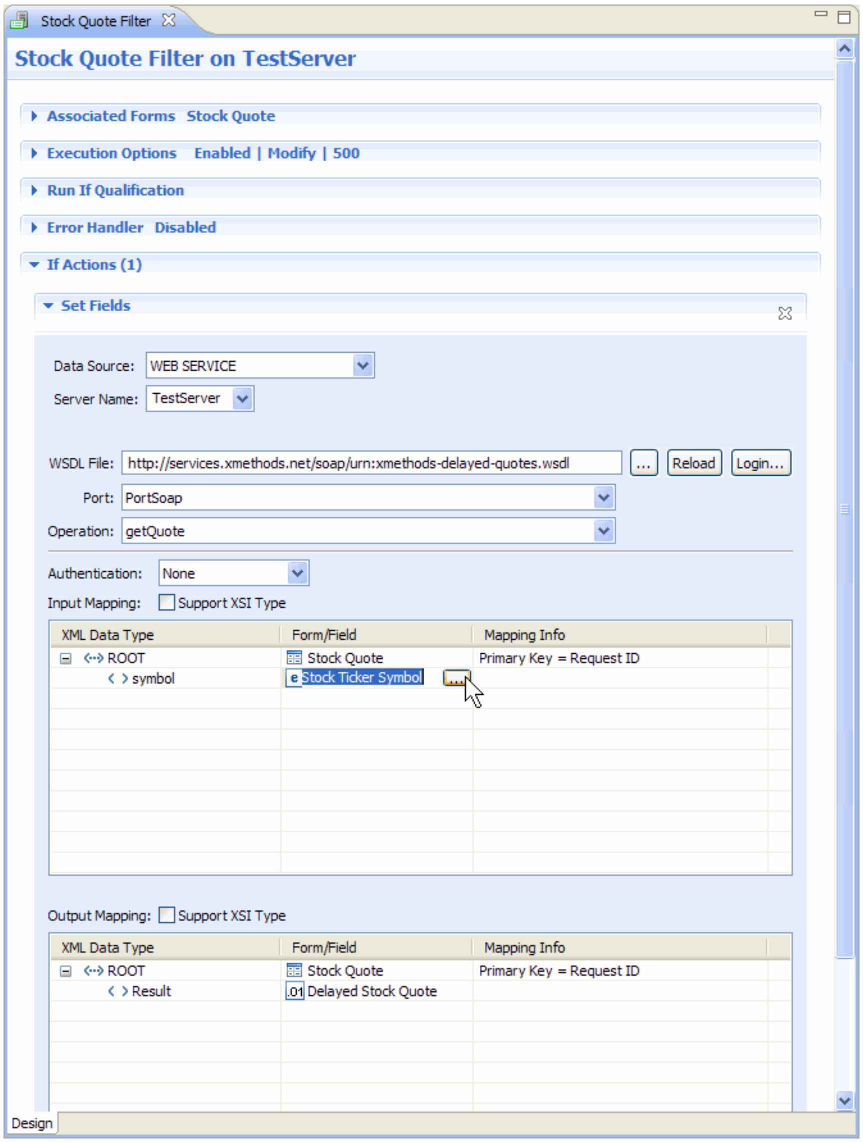
16 If prompted, enter the user name and password required by the remote web server for basic authentication.

17 From the Port drop-down menu, select the port for the web service.

18 From the Operation drop-down menu, select the `getQuote` operation.

The menu is automatically populated with the available operations for the web service. This example has only one operation.

Figure D-8: Stock Quote filter—If Actions and Set Fields panels


Stock Quote Filter

Stock Quote Filter on TestServer

Associated Forms Stock Quote

Execution Options Enabled | Modify | 500

Run If Qualification

Error Handler Disabled

If Actions (1)

Set Fields

Data Source: WEB SERVICE

Server Name: TestServer

WSDL File: http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl

Port: PortSoap

Operation: getQuote

Authentication: None

Input Mapping: Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	Stock Quote	Primary Key = Request ID
symbol	Stock Ticker Symbol	

Output Mapping: Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	Stock Quote	Primary Key = Request ID
Result	Delayed Stock Quote	

19 In the Input Mapping table, highlight the cell adjacent to the symbol element, as shown in Figure D-8.

Note that the ROOT element of the XML schema is automatically mapped to the Stock Quote form.

20 Click the ellipsis button.

21 Use the Field Selector to chose the Stock Ticker Symbol field, which you created in step 2.

The field name appears in the Form / Field column. The symbol element (string type) is now mapped to the Stock Ticker Symbol field (designed to contain a string).

22 In the Output Mapping table, highlight the cell adjacent to the Result element, as shown in Figure D-8.

23 Click the ellipsis button.

24 Use the Field Selector to chose the Delayed Stock Quote field, which you created in step 2.

The Result element (string type) is now mapped to the Stock Quote field (designed to contain a string).

25 Choose File > Save, and name your filter Stock Quote Filter.

26 Open BMC Remedy User.

27 Open a new Stock Quote form, and enter data in the required fields.

28 Save the Stock Quote form.

29 Search to open the form in Modify mode.

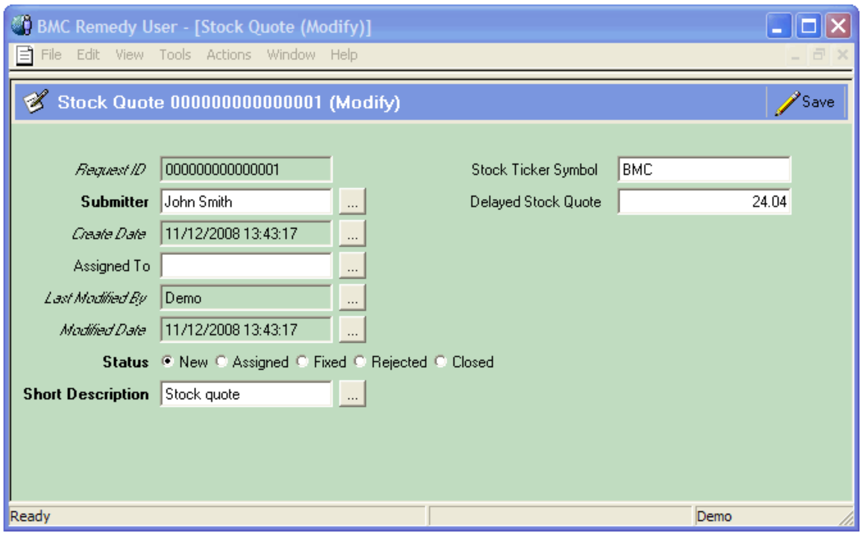
Recall that you created the Stock Quote Filter to execute on Modify (step 10).

30 Enter a symbol (for example, BMC) in the Stock Ticker Symbol field.

31 Click Save or Submit.

The quote for your stock is displayed in the Stock Ticker Symbol field (see Figure D-9).

Figure D-9: Stock Quote form in BMC Remedy User


BMC Remedy User - [Stock Quote (Modify)]
File Edit View Tools Actions Window Help
Stock Quote 000000000000001 (Modify)
Save

Request ID	000000000000001
Submitter	John Smith
Create Date	11/12/2008 13:43:17
Assigned To	
Last Modified By	Demo
Modified Date	11/12/2008 13:43:17
Status	[✓ New] Assigned Fixed Rejected Closed
Short Description	Stock quote
Stock Ticker Symbol	BMC
Delayed Stock Quote	24.04

Ready Demo

Example 3: Publishing a complex document

In this example, you publish a web service with two operations:

- `CreatePurchaseOrder` takes purchase order information as input and returns the purchase order ID as output.
- `GetPurchaseOrder` that takes the purchase order ID as input and returns information for that purchase order.

In this example, the process for publishing a complex document is as follows:

Step 1 Create an XML schema and save it as an XSD file (page 381).

Step 2 Create forms (page 382).

Step 3 Create a web service (page 386).

Step 4 Map the `CreatePurchaseOrder` operation (page 387).

Step 5 Map the `GetPurchaseOrder` operation (page 390).

Step 6 View your WSDL file (page 392).

▶ To create an XML schema

1 Create an XML schema containing the elements for your purchase order.

Here is a sample XML schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by
AR System -->
<xs:schema targetNamespace="http://tempuri.org" xmlns:xs="http://
www.w3.org/2001/XMLSchema" xmlns="http://tempuri.org"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="PO" type="PurchaseOrder">
    <xs:annotation>
       <xs:documentation>Comment describing your root element
       </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="PurchaseOrder">
    <xs:sequence>
       <xs:element name="POID" type="xs:string"/>
       <xs:element name="CompanyName" type="xs:string"/>
       <xs:element name="Description" type="xs:string"/>
       <xs:element name="PhoneNumber" type="xs:string"/>
       <xs:element ref="Items"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="LineItem">
       <xs:sequence>
          <xs:element name="ItemName" type="xs:string"/>
          <xs:element name="Quantity" type="xs:int"/>
```

```
            <xs:element name="ItemId" type="xs:string"/>
         </xs:sequence>
    </xs:complexType>
    <xs:element name="Item" type="LineItem"/>
    <xs:element name="Items">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="Item" minOccurs="0"
            maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

2 Name the schema `PurchaseOrder.xsd`.

▶ To create the forms

1 Create a form named Purchase Order.

This is the parent form.

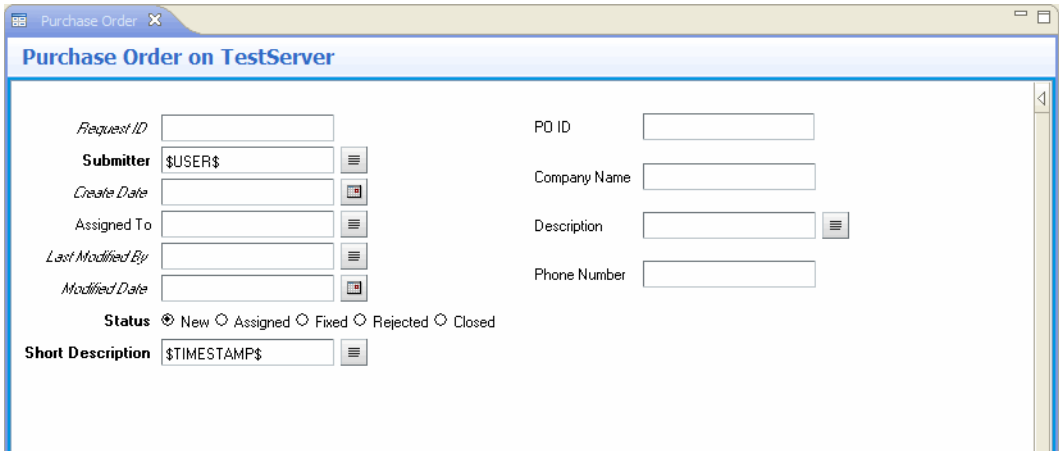
2 Add the following character fields to the form:

- PO ID
- Company Name
- Description
- Phone Number

3 Provide a default value for the Submitter and Short Description fields, because they are required fields.

Your form should be similar to the sample in Figure D-10.

Figure D-10: Purchase Order form in BMC Remedy Developer Studio



Request ID		PO ID	
Submitter	$USER$	Company Name	
Create Date		Description	
Assigned To		Phone Number	
Last Modified By			
Modified Date			
Status	[✓ New] Assigned Fixed Rejected Closed		
Short Description	$TIMESTAMP$		

4 (optional) Hide the core fields on the form (that is, all except those you created in step 2).

5 Choose Form > Form Properties.

6 On the Indexes page, click the New button to create a new index definition.

7 Click the Add button, choose the PO ID field, then click OK.

8 In the Unique Index column, change the Unique Index value to Yes.

9 Click OK to save your index definition.

NOTE

You must define an index for the PO ID field because PO ID is the primary key in this example. Usually, however, Request ID is the primary key. An index is predefined for the Request ID field, by default.

10 Save the Purchase Order form.

11 Create another form named Line Items.

This is the detail form.

12 Add the following character fields to the Line Items form:

- Line Item PO ID (the foreign key)
- Item ID (the distinguishing key)
- Item Name

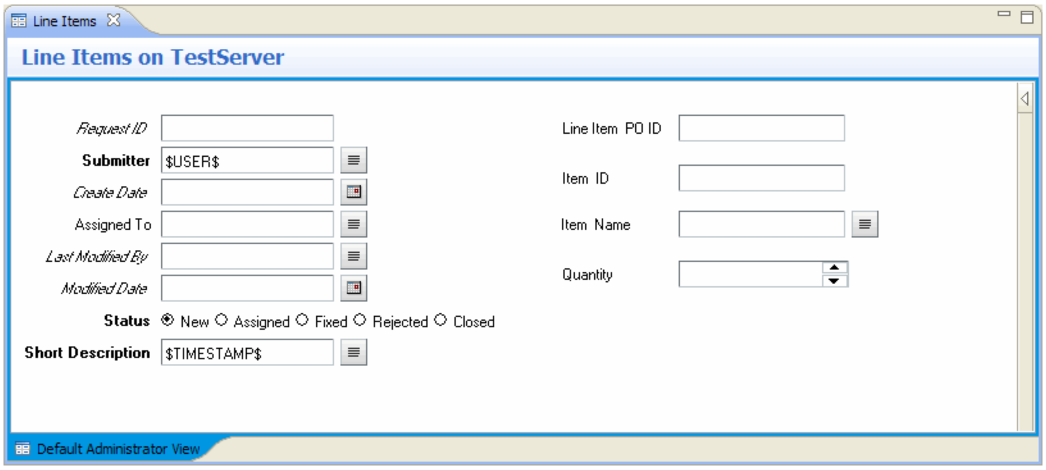
13 Add an integer field named Quantity to the form.

14 Set the Data Length of the Line Item PO ID and Item ID fields to 40.

15 Provide a default value for the Submitter and Short Description fields, because they are required fields.

Your form should be similar to the sample in Figure D-11.

Figure D-11: Line Items form in BMC Remedy Developer Studio



Request ID	
Submitter	$USER$
Create Date	
Assigned To	
Last Modified By	
Modified Date	
Status	[✓ New] Assigned Fixed Rejected Closed
Short Description	$TIMESTAMP$
Line Item PO ID	
Item ID	
Item Name	
Quantity	

16 (optional) Hide the core fields on the form (that is, all except those you created in step 12.)

17 Choose Form > Form Properties.

18 On the Indexes page, click the New button to create a new index definition.

19 Click the Add button.

20 Use the Field Selector to choose the Line Item PO ID and Item ID fields, and click OK.

21 In the Unique Index column, change the Unique Index value to Yes.

Making the distinguishing key and the foreign key unique keeps the data consistent, even if the data is submitted with other tools on the form.

22 Click OK to save your index definition.

23 Save the Line Items form.

NOTE
The following steps are optional. They enable you to view the line items of the corresponding purchase order in the same form.

24 Open the Purchase Order form.

25 Right-click a blank area of the form, then choose Create a New Field from the context menu.

26 Create a table field using the Table - List View option.

27 Select the field, then highlight the Tree/Table Property in the Properties tab.

28 Click the corresponding ellipsis button in the Value column of the Properties tab, as shown in Figure D-13.

The Tree/Table Property dialog box opens.

29 In the Tree/Table Property dialog box, choose the Line Items form (see Figure D-12).

30 In the Tree/Table Property dialog box, select the following fields:

- Item ID
- Item Name
- Quantity

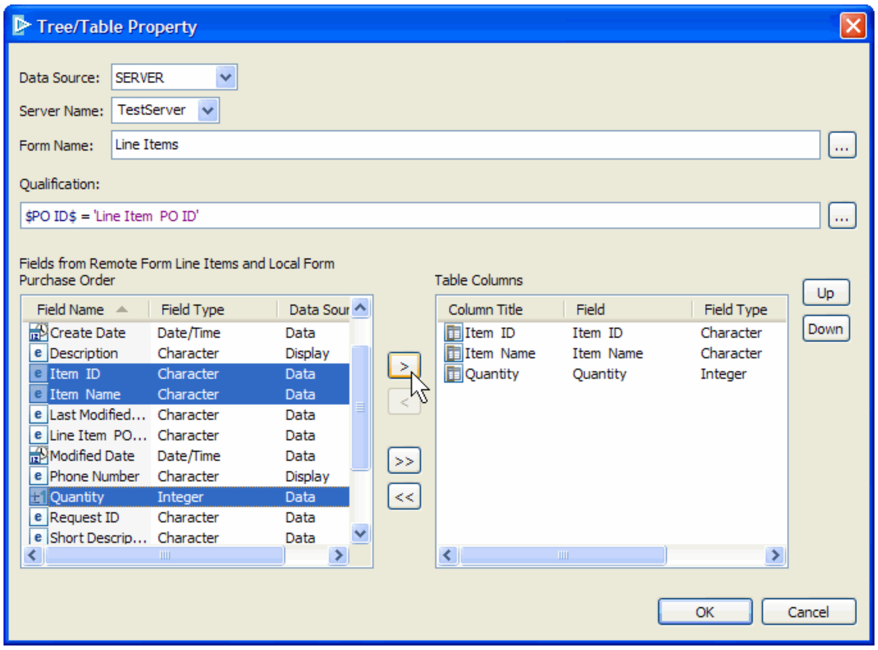
31 Click the right arrow button to add these fields to the list of table columns.

32 In the Qualification field, enter this qualification to make sure that PO ID from the parent form (Purchase Order) matches Line Item PO ID from the Line Items form:

```
$PO ID$ = 'Line Item PO ID'
```

NOTE
If Request ID is your primary key, enter this qualification instead:
`$Request ID$ = 'Line Item PO ID'`

Figure D-12: Field Properties dialog box—Tree/Table Property


Tree/Table Property

Data Source: SERVER

Server Name: TestServer

Form Name: Line Items

Qualification:

$PO ID$ = 'Line Item PO ID'

Fields from Remote Form Line Items and Local Form Purchase Order

Field Name	Field Type	Data Sour
Create Date	Date/Time	Data
Description	Character	Display
Item ID	Character	Data
Item Name	Character	Data
Last Modified...	Character	Data
Line Item PO...	Character	Data
Modified Date	Date/Time	Data
Phone Number	Character	Display
Quantity	Integer	Data
Request ID	Character	Data
Short Descrip...	Character	Data

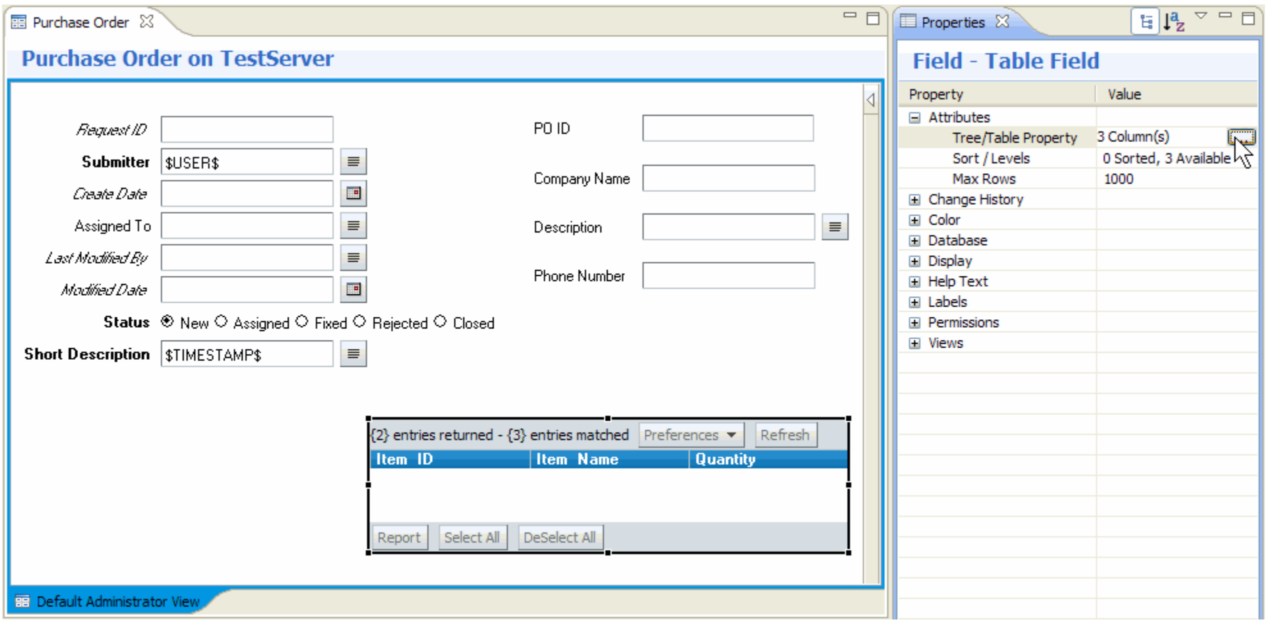
Table Columns

Column Title	Field	Field Type
Item ID	Item ID	Character
Item Name	Item Name	Character
Quantity	Quantity	Integer

33 Click OK to close the Tree / Table Property dialog box.

Your form should look similar to the sample in Figure D-13.

Figure D-13: Purchase Order form and Properties tab



Request ID	
Submitter	$USER$
Create Date	
Assigned To	
Last Modified By	
Modified Date	
Status	[✓ New] Assigned Fixed Rejected Closed
Short Description	$TIMESTAMP$
PO ID	
Company Name	
Description	
Phone Number	

{2} entries returned - {3} entries matched

Item ID	Item Name	Quantity

Property	Value
Attributes	
Tree/Table Property	3 Column(s)
Sort / Levels	0 Sorted, 3 Available
Max Rows	1000
Change History	
Color	
Database	
Display	
Help Text	
Labels	
Permissions	
Views	

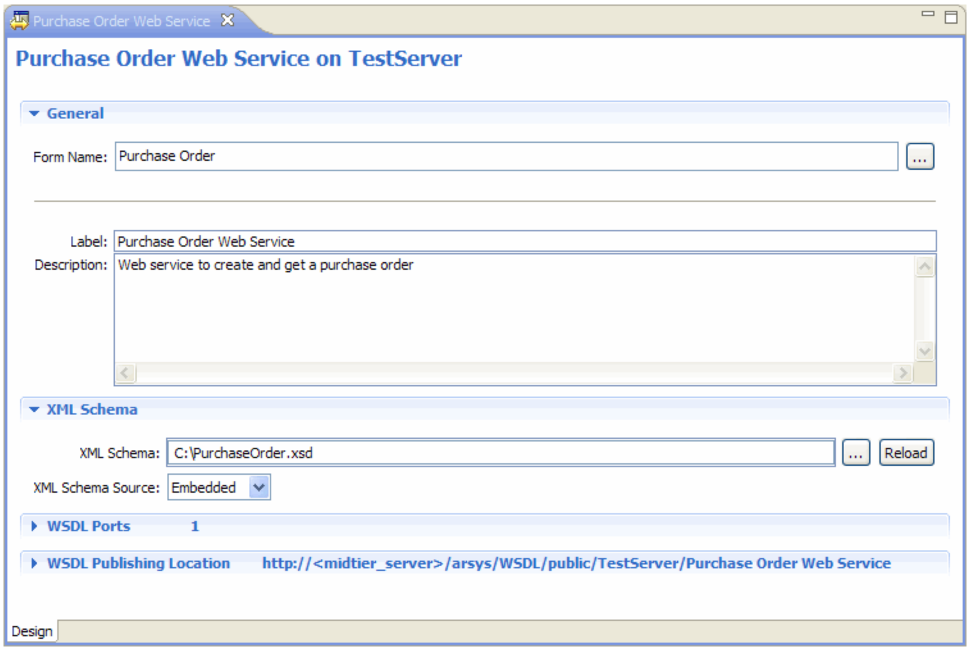
34 Save and close the Purchase Order form.

▶ To create your web service

1. In AR System Navigator, double-click the Forms object to display a list of forms on a tab.
2. In the Forms tab, right-click the Purchase Order form (see the example in Figure D-2 on page 373).
3. Choose Create Web Service.

   The web service appears in a new tab. AR System automatically fills in the Form Name field.
4. Right-click the General tab and choose Expand All Panels.
5. In the XML Schema field, enter a path or browse to `PurchaseOrder.xsd`, which you created at the beginning of this section.
6. Click the Reload button.
7. In the Label field, enter `Purchase Order Web Service`.
8. In the Description field, enter `Web service to create and get a purchase order`.
9. In the WSDL Operations panel, perform the following actions:
   a. Right-click the Set Operation panel, and click Remove.
   b. Right-click the Service Operation panel, and click Remove.
   c. Right-click the Get Operation panel for the operation called `GetList`, and click Remove.
10. Select `Create`.
11. In the Name field for the Create Operation, replace `Create` with `CreatePurchaseOrder`.
12. In the Name field for the Get Operation, replace `Get` with `GetPurchaseOrder`.
13. Choose File > Save, and name your web service Purchase Order Web Service.

Figure D-14: Purchase Order web service



Purchase Order Web Service on TestServer

General

Form Name: Purchase Order

Label: Purchase Order Web Service

Description: Web service to create and get a purchase order

XML Schema

XML Schema: C:\PurchaseOrder.xsd

XML Schema Source: Embedded

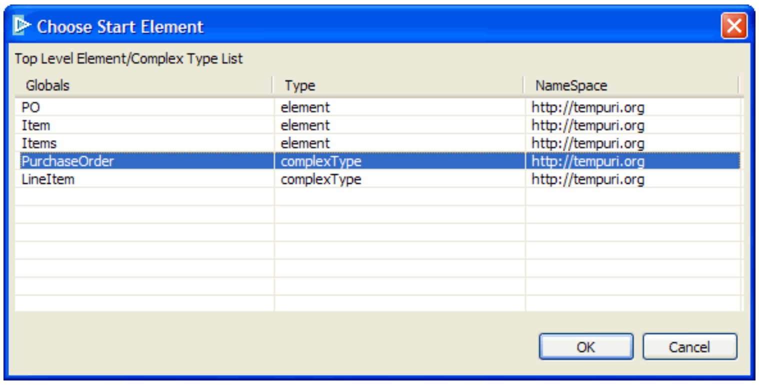
WSDL Ports 1

WSDL Publishing Location http://<midtier_server>/arsys/WSDL/public/TestServer/Purchase Order Web Service

▶ To map the CreatePurchaseOrder operation

1. Open the panel for the Create Operation called `CreatePurchaseOrder`.
2. Choose XML Schema from the Data Source Type drop-down menu, located above the Input Map.
3. Click the Choose button, then click OK to verify that you want to proceed with input mapping.
4. In the Choose Start Element dialog box, select PurchaseOrder and click OK.

Figure D-15: Choose start element dialog box



Choose Start Element

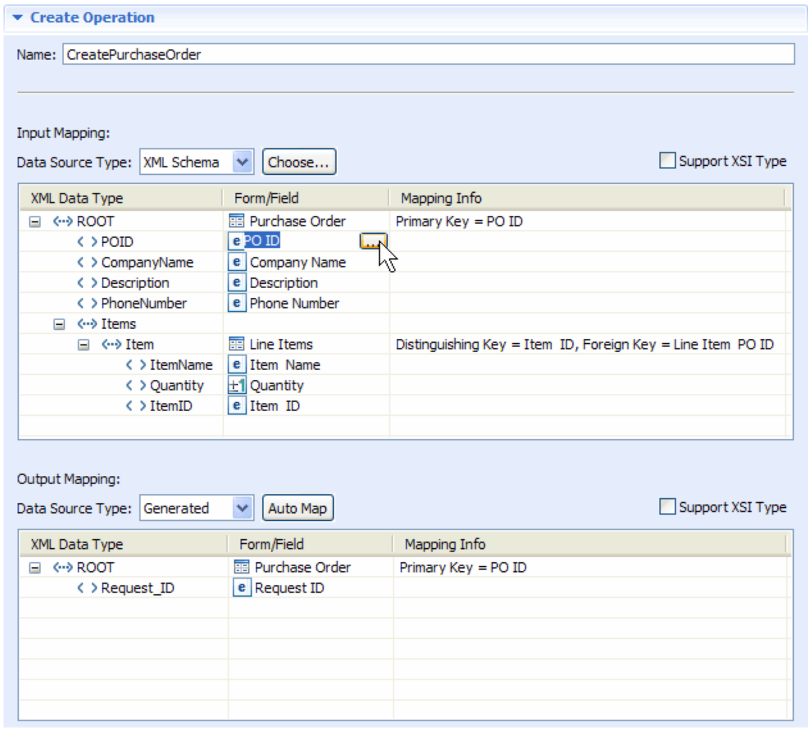
Top Level Element/Complex Type List

Globals	Type	NameSpace
PO	element	http://tempuri.org
Item	element	http://tempuri.org
Items	element	http://tempuri.org
PurchaseOrder	complexType	http://tempuri.org
LineItem	complexType	http://tempuri.org

5. Select the ROOT element in the XML Data Type column.

   Note that ROOT is already mapped to the Purchase Order form.

Figure D-16: Mapping for the CreatePurchaseOrder operation



Create Operation

Name: CreatePurchaseOrder

Input Mapping:

Data Source Type: XML Schema

Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	Purchase Order	Primary Key = PO ID
POID	PO ID	
CompanyName	Company Name	
Description	Description	
PhoneNumber	Phone Number	
Items		
Item	Line Items	Distinguishing Key = Item ID, Foreign Key = Line Item PO ID
ItemName	Item Name	
Quantity	Quantity	
ItemID	Item ID	

Output Mapping:

Data Source Type: Generated

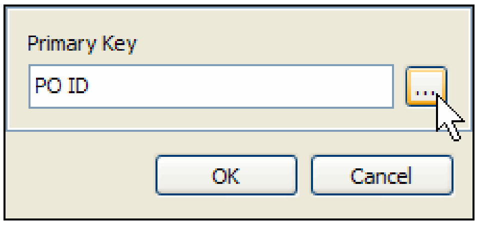
Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	Purchase Order	Primary Key = PO ID
Request_ID	Request ID	

6 Click Primary Key in the Mapping Info column.

7 Click the ellipsis button to open the Field Selector.

Figure D-17: Primary Key selection



Primary Key: PO ID

8 Choose PO ID as your primary key, then click OK twice.

9 Map the fields from the Purchase Order form as follows, referring to Figure D-16:

XML elements and complex types	Form and fields	Primary Key field
ROOT (PurchaseOrder)	Purchase Order	PO ID
POID	PO ID	
CompanyName	Company Name	
Description	Description	
PhoneNumber	Phone Number	

10 Under the Items element in the XML Data Type column, select the Item element.

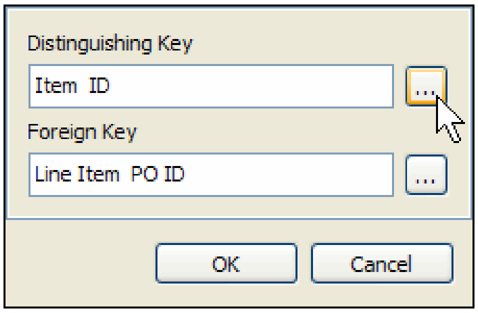
11 In the Form / Field column, click the empty table cell adjacent to the Item element.

12 Click the ellipsis button, then choose the Line Items form and click OK.

The Item element is now mapped to the Line Items form, as shown in Figure D-16.

13 Click Distinguishing Key / Foreign Key in the Mapping Info column.

Figure D-18: Distinguishing Key and Foreign Key selection



Distinguishing Key: Item ID

Foreign Key: Line Item PO ID

14 Click the ellipsis button adjacent to the Distinguishing Key field.

15 Use the Field Selector to choose the Item ID field, and click OK.

NOTE

The AR System does not allow the mapping of an integer as the distinguishing key during the consumption of a web service.

16 In the same manner, specify a Foreign Key value of Line Item PO ID.

17 Click OK to confirm your key selections.

18 Map the other fields from the Line Items form as follows, referring to Figure D-16:

XML elements and complex types	Form and fields	Distinguishing Key field	Foreign Key field
ROOT / Items / Item	Line Items	Item ID	Line Item PO ID
ItemName	Item Name		
Quantity	Quantity		
ItemID	Item ID		

19 For output mapping, choose Generated from the Data Source Type drop-down menu.

20 Click the Auto Map button, then click OK to verify that you want to proceed with output mapping.

The ROOT element in the XML Data Type column is mapped to the Purchase Order form.

21 In the Mapping Info column, change the Primary Key to PO ID (see Figure D-17).

22 Verify that the Request ID element is mapped to the Request ID field.

Your output mapping table should look similar to the one in Figure D-16.

23 In the Properties tab, assign Public permissions for the web service.

24 Click File > Save.

▶ To map the GetPurchaseOrder operation

1 Open the panel for the Get Operation called `GetPurchaseOrder`.

2 Choose Generated from the Data Source Type drop-down menu, located above the Input Map.

3 Click the Auto Map button.

4 Click OK to verify that you want to proceed with input mapping, then choose the standard Get operation mapping option.

The input map should contain two XML elements, by default—ROOT and Request_ID.

5 Click the Request_ID element, and change the name to POID.

Your input mapping table should look similar to the one in Figure D-19.

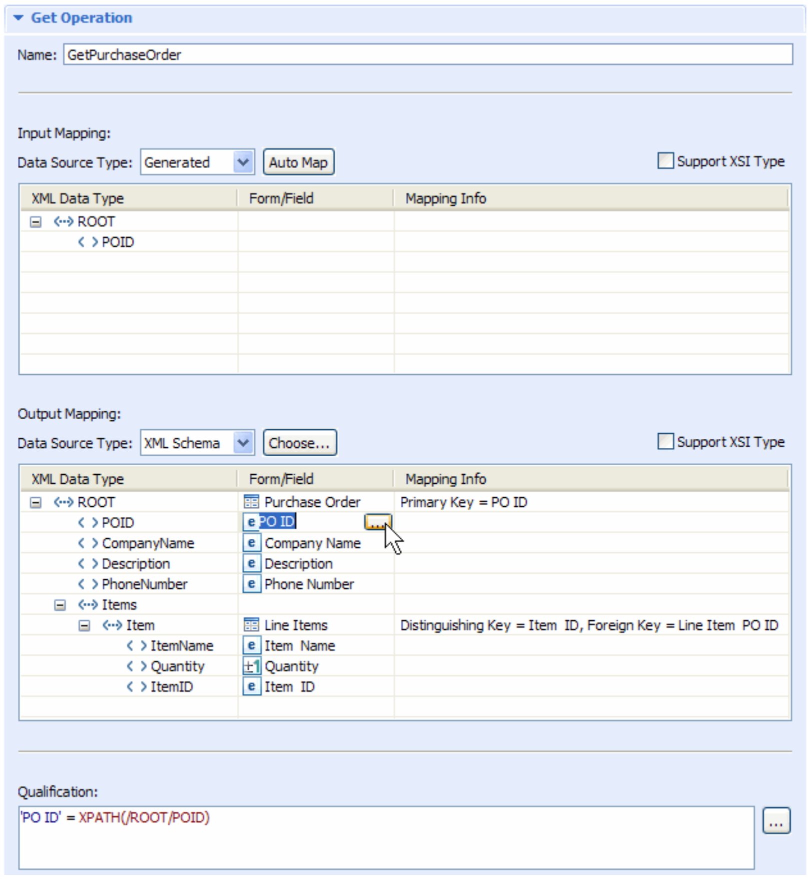
6 In the Qualification field, replace the qualification with this text:

```
'PO ID' = XPATH(/ROOT/POID)
```

TIP

You can type or paste the text, or you can use the Expression Editor to build the qualification. To open the Expression Editor, click the ellipsis button.

Figure D-19: Mapping for the GetPurchaseOrder operation



Get Operation

Name: GetPurchaseOrder

Input Mapping:

Data Source Type: Generated | Auto Map | Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT		
POID		

Output Mapping:

Data Source Type: XML Schema | Choose... | Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	Purchase Order	Primary Key = PO ID
POID	PO ID	
CompanyName	Company Name	
Description	Description	
PhoneNumber	Phone Number	
Items		
Item	Line Items	Distinguishing Key = Item ID, Foreign Key = Line Item PO ID
ItemName	Item Name	
Quantity	Quantity	
ItemID	Item ID	

Qualification:

'PO ID' = XPATH(/ROOT/POID)

7 For output mapping, choose XML Schema from the Data Source Type drop-down menu.

8 Click the Choose button, then click OK to verify that you want to proceed with input mapping.

9 In the Choose Start Element dialog box, select PurchaseOrder and click OK (see Figure D-15 on page 387).

10 Select the ROOT element in the XML Data Type column.

Note that ROOT is already mapped to the Purchase Order form.

11 Click Primary Key in the Mapping Info column.

12 Click the ellipsis button to open the Field Selector (see Figure D-17 on page 388).

13 Choose PO ID as your primary key, then click OK twice.

14 Map the fields from the Purchase Order form as follows, referring to Figure D-19:

XML elements and complex types	Form and fields	Primary Key field
ROOT	Purchase Order	PO ID
POID	PO ID	
CompanyName	Company Name	

XML elements and complex types	Form and fields	Primary Key field
Description	Description	
PhoneNumber	Phone Number	

15 Under the Items element in the XML Data Type column, select the Item element.

16 In the Form / Field column, click the empty table cell adjacent to the Item element.

17 Click the ellipsis button, then choose the Line Items form and click OK.

The Item element is now mapped to the Line Items form, as shown in Figure D-19.

18 Click Distinguishing Key / Foreign Key in the Mapping Info column (see Figure D-18 on page 389).

19 Click the ellipsis button adjacent to the Distinguishing Key field.

20 Use the Field Selector to choose the Item ID field, and click OK.

NOTE

The AR System does not allow the mapping of an integer as the distinguishing key during the consumption of a web service.

21 In the same manner, specify a Foreign Key value of Line Item PO ID.

22 Click OK to confirm your key selections.

23 Map the fields from the Line Items form as follows, referring to Figure D-19:

XML elements and complex types	Form and fields	Distinguishing Key field	Foreign Key field
ROOT / Items / Item	Line Items	Item ID	PO ID
ItemName	Item Name		
Quantity	Quantity		
ItemID	Item ID		

24 In the Properties tab, assign Public permissions for the web service.

25 Save the web service.

▶ To view the WSDL for the Purchase Order web service

1 Verify that your web service has public permissions.

2 Expand the WSDL Publishing Location panel.

A sample URL for your WSDL file is displayed in the Specify mid-tier's WSDL handler URL field (see Figure D-20).

3 Modify the URL appropriately for your configuration:

a Replace `<midtier_server>` with the name of the web server where the mid tier is running.

b Add `/public` after WSDL.

For example, if the mid tier server is `TestServer`, you would use this URL:

```
http://TestServer/arsys/WSDL/public/TestServer/Purchase Order Web
Service
```

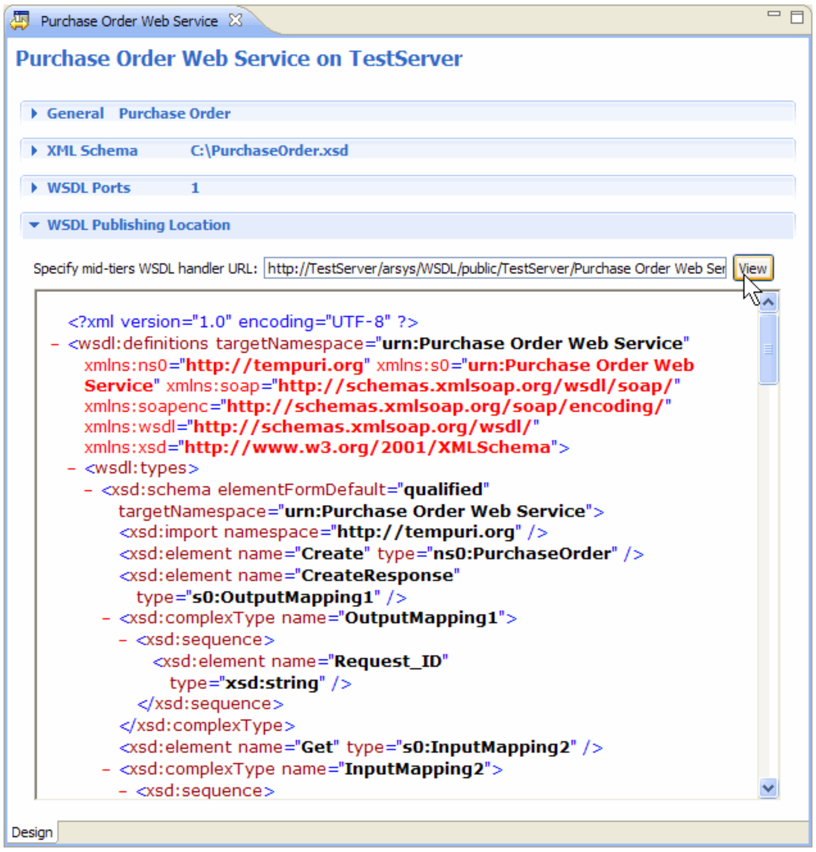
4 Click File > Save.

5 Click View to display the content of the WSDL file.

6 If prompted, enter a user name and password and click Login.

The WSDL code is displayed, as shown in Figure D-20.

Figure D-20: WSDL file for the Purchase Order web service



Purchase Order Web Service on TestServer

General	Purchase Order
XML Schema	C:\PurchaseOrder.xsd
WSDL Ports	1
WSDL Publishing Location	

Specify mid-tiers WSDL handler URL: http://TestServer/arsys/WSDL/public/TestServer/Purchase Order Web Ser

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="urn:Purchase Order Web Service"
    xmlns:ns0="http://tempuri.org" xmlns:s0="urn:Purchase Order Web
    Service" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <xsd:schema elementFormDefault="qualified"
        targetNamespace="urn:Purchase Order Web Service">
        <xsd:import namespace="http://tempuri.org" />
        <xsd:element name="Create" type="ns0:PurchaseOrder" />
        <xsd:element name="CreateResponse"
          type="s0:OutputMapping1" />
      - <xsd:complexType name="OutputMapping1">
        - <xsd:sequence>
            <xsd:element name="Request_ID"
              type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="Get" type="s0:InputMapping2" />
      - <xsd:complexType name="InputMapping2">
        - <xsd:sequence>
```

Example 4: Consuming a complex document

In this example, you access the Purchase Order web service you created in Example 3, and use the Set Fields filter action to get a purchase order and a line item record.

NOTE

Before testing this example, verify that your web service has public permissions.

In this example, the process for consuming a complex document is as follows:

Step 1 Create the forms (see page 394).

Step 2 Create the input mappings (see page 397).

Step 3 Create the output mappings (see page 398).

Step 4 Consume the web service (see page 399).

Step 5 View and verify the web service (see page 400).

For information about setting your environment to consume a web service created on the same AR System server, see "Managing web service performance issues" on page 94.

▶ To create your forms

1 Create two forms by following step 1 through step 34 in Example 3, but name the forms as follows:

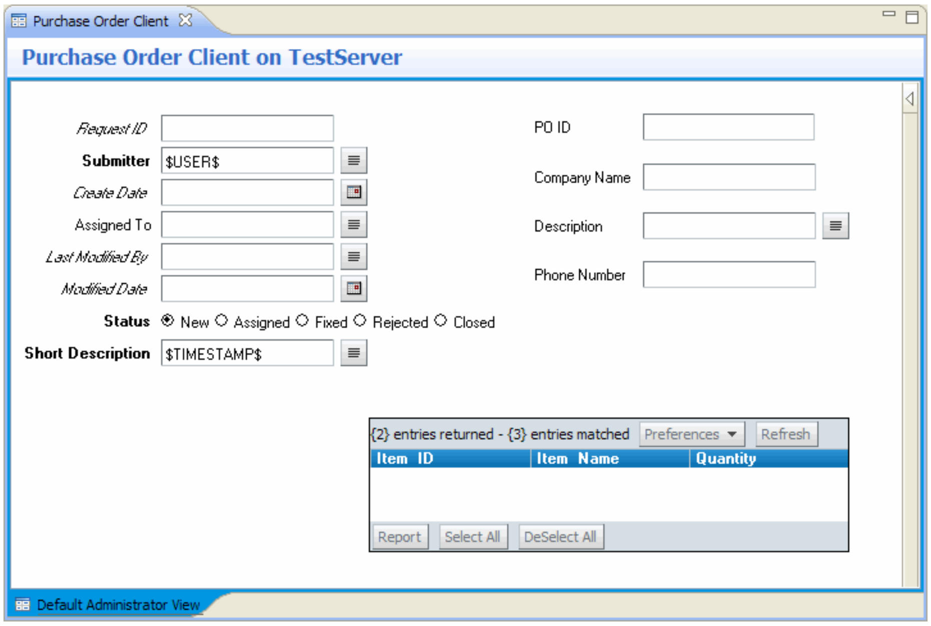
- Purchase Order Client
- Line Items Client

TIP

To copy a form, open it and choose File > Save As.

The Purchase Order Client form should be similar to the form in Figure D-21.

Figure D-21: Purchase Order Client form



Purchase Order Client

Purchase Order Client on TestServer

Request ID	
Submitter	$USER$
Create Date	
Assigned To	
Last Modified By	
Modified Date	
Status	[✓ New] Assigned Fixed Rejected Closed
Short Description	$TIMESTAMP$
PO ID	
Company Name	
Description	
Phone Number	

{2} entries returned - {3} entries matched Preferences Refresh

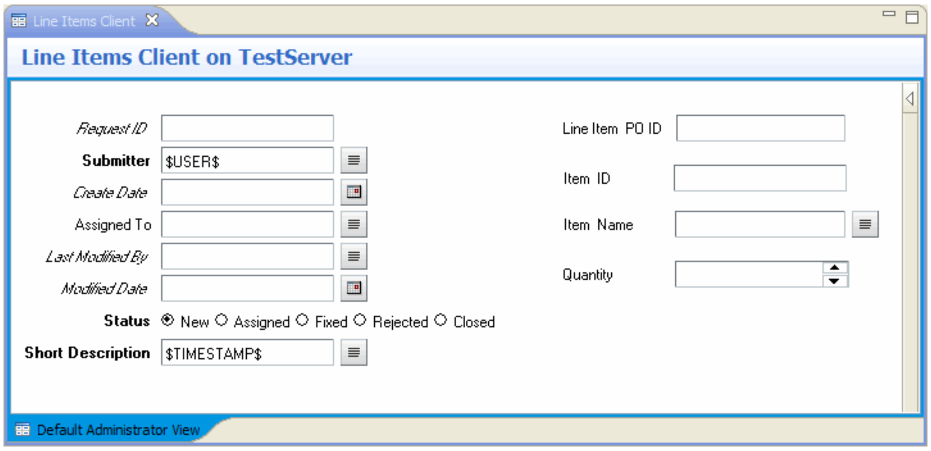
Item ID	Item Name	Quantity

Report Select All DeSelect All

Default Administrator View

The Line Items Client form should be similar to the form in Figure D-22.

Figure D-22: Line Items Client form



Line Items Client

Line Items Client on TestServer

Request ID	
Submitter	$USER$
Create Date	
Assigned To	
Last Modified By	
Modified Date	
Status	[✓ New] Assigned Fixed Rejected Closed
Short Description	$TIMESTAMP$
Line Item PO ID	
Item ID	
Item Name	
Quantity	

Default Administrator View

2 Choose File > New > Filter.

3 Select the server on which you want to create the filter, and click Finish.

4 Right-click the Associated Forms panel, then choose Expand All Panels.

5 In the Associated Forms panel, click the Add button (see Figure D-7).

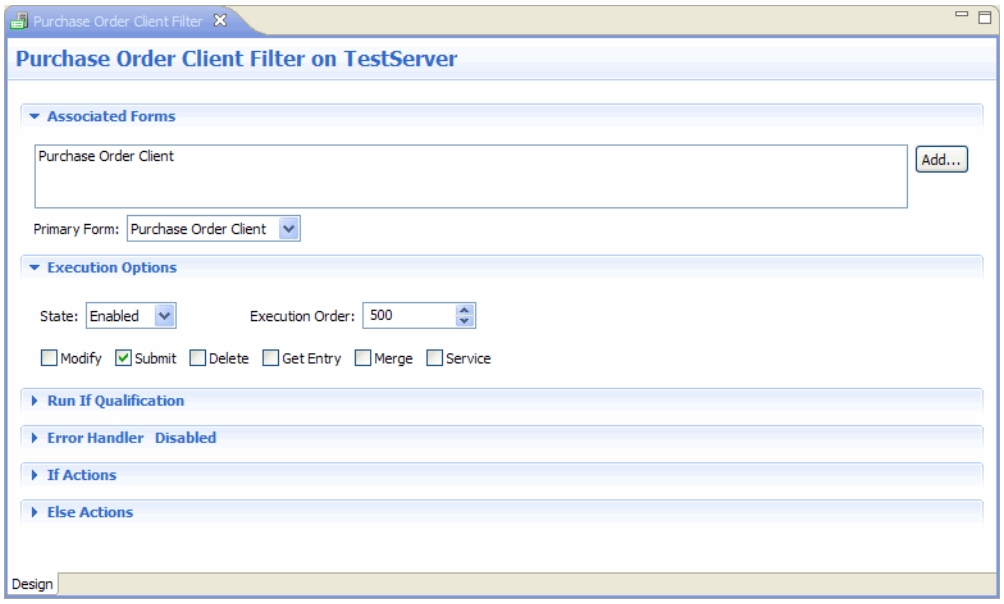
6 In the Form Selector dialog box, select the Purchase Order Client form to use as the base form, and click OK.

7 From the State list, choose Enabled.

8 In the Execution Order field, enter the execution order for the filter, or accept the default value.

9 In the Execution Options panel, select Submit.

Figure D-23: Purchase Order Client filter—basic options



Purchase Order Client Filter on TestServer

Associated Forms

Purchase Order Client

Primary Form: Purchase Order Client

Execution Options

State: Enabled   Execution Order: 500

☐ Modify ☑ Submit ☐ Delete ☐ Get Entry ☐ Merge ☐ Service

10 Right-click the If Actions panel, choose Add Action, and select Set Fields.

A new panel appears, where you configure the Set Fields action (see Figure D-24.)

11 From the Data Source list, choose WEB SERVICE.

12 From the Server Name list, select the server on which to store the web service mappings as a server object.

13 In the WSDL field, enter the path or browse to your WSDL file.

In this example, your URL might look like this:

```
http://TestServer/arsys/WSDL/public/TestServer/Purchase Order Web
Service
```

You can also enter a path to a local WSDL file. For example:

```
C:\Temp\Purchase Order Web Service.wsdl
```

14 Click Reload.

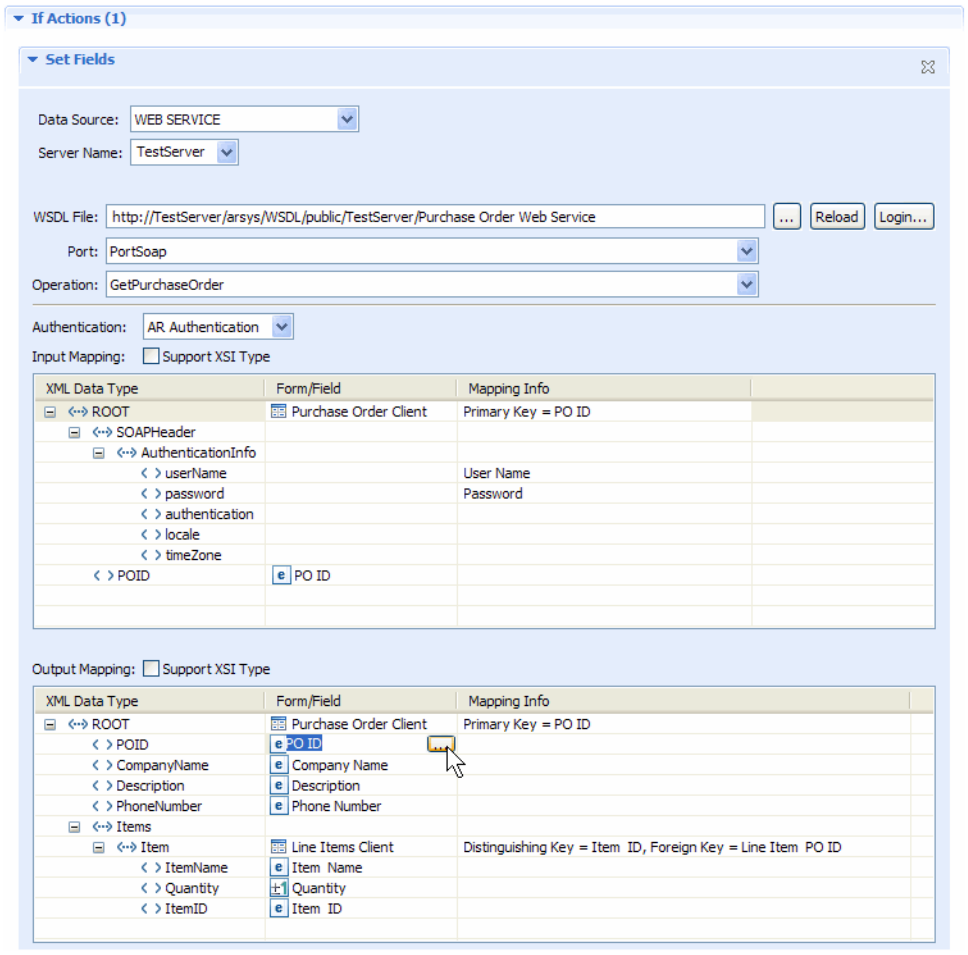
15 If prompted, enter the user name and password required by the remote web server for basic authentication.

16 From the Port drop-down menu, select the port for the web service.

17 From the Operation drop-down menu, select the `getPurchaseOrder` operation.

The menu is automatically populated with the available operations for the web service.

Figure D-24: Purchase Order Client filter—Set Fields action



If Actions (1)

Set Fields

Data Source: WEB SERVICE

Server Name: TestServer

WSDL File: http://TestServer/arsys/WSDL/public/TestServer/Purchase Order Web Service

Port: PortSoap

Operation: GetPurchaseOrder

Authentication: AR Authentication

Input Mapping: ☐ Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	Purchase Order Client	Primary Key = PO ID
SOAPHeader		
AuthenticationInfo		
userName		User Name
password		Password
authentication		
locale		
timeZone		
POID	PO ID	

Output Mapping: ☐ Support XSI Type

XML Data Type	Form/Field	Mapping Info
ROOT	Purchase Order Client	Primary Key = PO ID
POID	PO ID	
CompanyName	Company Name	
Description	Description	
PhoneNumber	Phone Number	
Items		
Item	Line Items Client	Distinguishing Key = Item ID, Foreign Key = Line Item PO ID
ItemName	Item Name	
Quantity	Quantity	
ItemID	Item ID	

▶ To create input mappings

1 In the Input Mapping table, select the ROOT element in the XML Data Type column.

Note that ROOT is already mapped to the Purchase Order Client.

2 Click Primary Key in the Mapping Info column.

3 Click the ellipsis button to open the Field Selector (see Figure D-17 on page 388).

4 Choose PO ID as your primary key, then click OK twice.

5 Select the POID element in the XML Data Type column.

6 Click the adjacent cell in the Forms / Field column, then click the ellipsis button.

7 Choose the PO ID field and click OK.

Your input mapping should look similar to the one in Figure D-24.

8 Choose File > Save, and name your filter Purchase Order Client Filter.

▶ To create output mappings

1 Select the ROOT element in the XML Data Type column.

Note that ROOT is already mapped to the Purchase Order Client form.

2 Click Primary Key in the Mapping Info column.

3 Click the ellipsis button to open the Field Selector (see Figure D-17 on page 388).

4 Choose PO ID as your primary key, then click OK twice.

5 Map the fields from the Purchase Order Client form as follows, referring to Figure D-24:

XML elements and complex types	Form and fields	Primary Key field
ROOT	Purchase Order Client	PO ID
POID	PO ID	
CompanyName	Company Name	
Description	Description	
PhoneNumber	Phone Number	

6 Under the Items element in the XML Data Type column, select the Item element.

7 In the Form / Field column, click the empty table cell adjacent to the Item element.

8 Click the ellipsis button, then choose the Line Items Client form and click OK.

The Item element is now mapped to the Line Items Client form, as shown in Figure D-24.

9 Click Distinguishing Key / Foreign Key in the Mapping Info column (see Figure D-18 on page 389).

10 Click the ellipsis button adjacent to the Distinguishing Key field.

11 Use the Field Selector to choose the Item ID field, and click OK.

NOTE

The AR System does not allow the mapping of an integer as the distinguishing key during the consumption of a web service.

12 In the same manner, specify a Foreign Key value of Line Item PO ID.

13 Click OK to confirm your key selections.

14 Map the fields from the Line Items form as follows, referring to Figure D-19:

XML elements and complex types	Form and fields	Distinguishing Key field	Foreign Key field
ROOT / Items / Item	Line Items Client	Item ID	PO ID
ItemName	Item Name		
Quantity	Quantity		
ItemID	Item ID		

Your output mapping should look similar to the one in Figure D-24.

15 Save your changes to the filter.

▶ To consume the web service in an AR system client

1 Log in to BMC Remedy User.

2 Open a new Purchase Order form.

To create the Purchase Order form, see “Example 3: Publishing a complex document” on page 381.

3 Enter data in the required fields.

4 In the PO ID field, enter 111.

5 Save the record.

6 Open a new Line Items form.

To create the Line Items form, see “Example 3: Publishing a complex document” on page 381.

7 Enter data in the required fields.

8 In the Line Item PO ID field, enter 111.

9 Save the form.

10 Open a new Purchase Order Client form.

11 In the PO ID field, enter 111.

12 Save the form.

Recall that the Purchase Order Client filter executes on Submit.

13 In the Purchase Order Client form, open the request you created in step 12.

If the `GetPurchaseOrder` web service was successful, the request is submitted and contains the same values as the record in the Purchase Order form where you set PO ID to 111.

The values corresponding to the request you created for the Line Items form (step 6–step 9) also appear in the Line Items Client form.

▶ To view and verify your web service

1. Stop the AR System server.
2. Open the `armonitor.cfg` file.

   The default location of this file is `C:\Program Files\AR System\Conf`.
3. Locate the following line (or a similar one), and comment it out:

   ```
   "C:\Program Files\AR System\arplugin.exe" -i "C:\Program Files\
     AR System\" -m
   ```
4. Copy the line, and run it from the command line.
5. Verify that the following message is displayed:

   ```
   Loaded Web Services plugin properly
   ```

Appendix

E Adding a certificate to a certificate database

This appendix explains how to add a certificate to a certificate database (`cert8.db` file).

The following topics are provided:

- About certificate databases (page 402)
- Creating a certificate database (page 402)
- Adding a certificate to a certificate database (page 403)
- Listing certificates in a certificate database (page 404)

NOTE

AR System release 7.5.00 and higher, use Mozilla Network Security Services (NSS) 3.11.4, which also supports `cert7.db` files and automatically converts a `cert7.db` file to a `cert8.db` file. If you have a `cert7.db` certificate database, you do not need to upgrade to `cert8.db`.

About certificate databases

AR System uses the Mozilla C-LDAP libraries to support LDAP plug-ins and remote authentication. These libraries enable LDAP plug-ins to use NSS to establish Secure Sockets Layer (SSL) connections with LDAP servers. To do this, NSS requires the LDAP server's certification authority (CA) certificate to be in a certificate database (`cert8.db` file).

NOTE

This information is provided as a convenience only. BMC assumes that customers have a working certificate database.

When you configure LDAP plug-ins that use SSL connections, you must specify the directory that contains the certificate database. See these procedures:

- "To configure the ARDBC LDAP plug-in" on page 137
- "To configure settings for the AREA LDAP plug-in" on page 146

The sections in this appendix explain how to

- Create a certificate database (page 402)
- Add a certificate to a certificate database (page 403)
- List the certificates in a certificate database (page 404)

NOTE

To perform the procedures in this appendix, use the command-line `certutil` utility, which is included in the Mozilla NSS security tools set (see `http://www.mozilla.org/projects/security/pki/nss/tools/`).

Creating a certificate database

If you do not have a certificate database (`cert8.db` file), create one on the AR System server by using the `certutil` utility as follows.

▶ To create a certificate database

At the command line, enter this command:

```
certutil -N -d certDir
```

In this command, *`certDir`* is the directory in which the certificate database is created.

Adding a certificate to a certificate database

Add a certificate to a certificate database (`cert8.db` file) by using the `certutil` utility as follows.

NOTE

The certificate that you want to add must exist on your LDAP server. For information about creating certificates and adding them to your LDAP server, see your LDAP server documentation.

▶ To add a certificate to a certificate database

At the command line, enter this command:

```
certutil -A -n nickname -t "trustedAttributes" -d certDir -i certName.cer
```

In this command:

- `-A` adds an existing certificate to a certificate database.

  The certificate database should exist. If it does not, this option creates one.

- `-n` *`nickname`* specifies the nickname of the certificate.

  If the nickname contains spaces, enclose it in double quotation marks.

- `-t "`*`trustedAttributes`*`"` specifies the trusted attributes to apply to a certificate when adding it to a certificate database.

  Each certificate includes three trust categories: SSL, email, and object signing. For each category, specify zero or more of these trust attribute codes:

Code	Description
`p`	The client or server is a valid peer.
`P`	The client or server is a trusted peer (implies `p`).
`c`	The certificate was issued by a valid CA.
`T`	This CA is trusted to issue client certificates (implies `c`).
`C`	(SSL only) This CA is trusted to issue server certificates (implies `c`).
`u`	This certificate can be used for authentication or signing.
`w`	Send a warning. This attribute is used with another attribute to include a warning when the certificate is used in the context of that attribute.

  Use commas to separate each category. Enclose the entire set of attributes in double quotation marks. For example, this is a standard trust attribute configuration: `"PTCu,P,P"`.

- `-d` *`certDir`* specifies the directory that contains the `cert8.db` and `key3.db` files (these files must reside in the same directory).

  The specified directory must exist. If it does not, the command fails.

- `-i` *`certName`*`.cer` specifies the file name of the certificate to add to the certificate database.

Listing certificates in a certificate database

List the certificates in a certificate database (`cert8.db` file) by using the `certutil` utility as follows.

▶ To list all the certificates in a cert8.db file in a specified directory

At the command line, enter this command:

```
certutil -L -d certDir
```

where *`certDir`* specifies the directory that contains the `cert8.db` file.

Example

```
C:\conf>certutil -L -d cert
QAtest                                          CT,P,P
```

▶ To list information about a specified certificate

At the command line, enter this command:

```
certutil -L -d certDir -n nickname
```

where *`certDir`* specifies the directory that contains the certificate database.

- *`nickname`* is the nickname of the certificate whose information you want to list.
- *`certDir`* is the directory in which the certificate database that contains the certificate resides.

Example

```
C:\conf>certutil -L -d cert -n QAtest
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      0e:71:a2:de:d6:12:33:94:49:e9:2e:9d:3c:89:9b:54
    Signature Algorithm: PKCS #1 SHA-1 With RSA Encryption
    Issuer: "CN=bmc,DC=bmc,DC=com"
    Validity:
      Not Before: Wed Oct 15 10:08:36 2008
      Not After : Tue Oct 15 10:17:24 2013
    Subject: "CN=bmc,DC=bmc,DC=com"
    Subject Public Key Info:
      Public Key Algorithm: PKCS #1 RSA Encryption
      RSA Public Key:
        Modulus:
          94:10:62:3c:b0:c6:65:99:b8:b3:ed:60:38:4c:1a:48:
          40:3d:19:47:b5:0e:11:ac:57:f7:a9:c4:1b:48:4e:07
        Exponent: 65537 (0x10001)
    Signed Extensions:
      Name: Microsoft Enrollment Cert Type Extension
      Data: "CA"
```

```
            Name: Certificate Key Usage
            Usages: Digital Signature
              Certificate Signing
              CRL Signing

            Name: Certificate Basic Constraints
            Critical: True
            Data: Is a CA with no maximum path length.

            Name: Certificate Subject Key ID
            Data:
              ef:24:ae:81:c9:95:f6:e0:97:8c:38:f0:d4:66:8e:75:
              78:89:fb:18

            Name: CRL Distribution Points
            URI: "ldap:///CN=bmc,CN=QAtest,CN=CDP,CN=Public%20Key
              %20Services,CN=Services,CN=Configuration,DC=bmc,DC=com?
              certificateRevocationList?base?objectClass=cRLDistribution
              Point"

            Name: Microsoft CertServ CA version
            Data: 0 (0x0)

        Signature Algorithm: PKCS #1 SHA-1 With RSA Encryption
        Signature:
          03:21:71:3e:00:0b:37:10:8f:27:df:d9:92:d7:ef:6f:
          7c:af:db:c5:f5:4f:54:e3:9b:00:46:21:89:25:c8:2f

        Fingerprint (MD5):
          26:4E:87:0E:AE:8C:23:67:11:AF:90:75:C8:C6:8F:A7
        Fingerprint (SHA1):
          4C:D5:BF:57:78:DF:91:F8:23:BC:9C:95:A8:39:74:21:6A:50:D8:A7

        Certificate Trust Flags:
          SSL Flags:
            Valid CA
            Trusted CA
            Trusted Client CA
          Email Flags:
            Valid Peer
            Trusted
          Object Signing Flags:
            Valid Peer
            Trusted
```

Appendix

F AR System change ID utility

Beginning with version 7.6.04, AR System supports the AR System change ID utility, which enables you to change the IDs of certain objects. This section explains the purpose of the utility and describes its usage.

The following topics are provided:

- Introduction to archgid (page 408)
- Syntax of archgid (page 412)
- Using archgid in prompt-driven mode (page 415)
- Using archgid in bulk mode (page 417)

Introduction to archgid

The AR System change ID utility (commonly called `archgid`) enables you to change the ID of a form, field, view, or group. It serves the following purposes:To synchronize IDs for the same form across multiple servers

- To control the ID if needed for direct SQL access.
- To change IDs of objects that are within BMC reserved ID ranges *before* converting your pre-7.6.04 extensions and customizations to overlays and custom objects

  In most cases, you would need to change view IDs and field IDs. See the *Preserving Customizations with Overlays* document, “BMC reserved ID ranges” and “Comparing objects, identifying nonpermitted modifications, and fixing them.”

File name, version, and location

The utility file is located at *`ARSystemServerInstallDir`*`/ARSystem`, and is named as follows:

- Windows—`archgid.exe`
- UNIX—`archgid`

Compatibility

The `archgid` utility performs direct SQL database updates at a fundamental level with the data dictionary. Before performing any updates the utility checks whether the AR System database version is compatible. If the database version is incompatible, it displays an error message.

The `archgid` utility is compatible with the following AR System databases:

- AR System server 5.x releases—5.0, 5.0.xx, 5.1, and 5.1.xx
- AR System server 6.x releases—6.0, 6.0.01, and 6.3
- AR System server 7.x releases—7.0, 7.0.01, 7.1, 7.5, 7.6.02, 7.6.03, and 7.6.04

Execution modes

You can execute the `archgid` utility in any the following modes or a combination of these:

- Prompt-driven mode—To execute the utility in this mode, enter `archgid` at the command line and press Enter. The utility displays the available options sequentially, and prompts you for input until it has gathered all the required information.

TIP

Do not use this mode when fixing numerous non-permitted customizations on field IDs and view IDs.

- Command-line mode—To execute the utility in this mode, enter `archgid` followed by the required parameters and their values at the command line and press Enter. If you forget to provide a required parameter or its value, the utility prompts you for that input.

Updates to database, references, or associated objects

For each type of object ID that the `archgid` utility changes, it performs the following updates to the database or the AR System object definitions:

Object type	Corresponding updates
Form	The following items are updated: ■ ID in all data dictionary tables, including those in which the form is used in join forms and object relationship references ■ Name of the data tables for the form All database views are rebuilt on the new table names. Note: When changing form ID references in workflow, the workflow object is considered connected only to the primary form referenced in shared workflow definitions. The utility changes only the primary form's ID, because that is considered to be the template or master form for the shared definition.
View	The following items are updated: ■ ID in all data dictionary tables ■ ID in all HTML / JSP web page definitions

Object type	Corresponding updates
Field	■ The name of the data columns for the field (if there is a database table or view) is updated, and all database views are rebuilt on the new column names. ■ The following instances of the old ID are updated: ■ ID in all data dictionary tables, including those in which the field is referenced in join forms, table or column references, query style menus, page fields, and object relationship references ■ ID in deployable application data form qualifications ■ ID in linked Archive forms and the qualification for those forms ■ ID in all HTML and JSP web page definitions ■ ID in all active link, filter, and escalation definitions, including references in special Run Process commands ■ ID in the AR System Message Catalog form if there is localized Help Text ■ ID in Distributed Mapping definitions, including DSO qualification, mapping, and return mapping ■ ID in flashboards variable definitions ■ The following references to the old ID are updated: ■ References in menus to the current form are noted as warnings of possible issues but are not changed, because the reference may or may not be to the field because a menu definition is not associated with a form. ■ References in macros embedded in active links are noted as warnings of possible issues but are not changed, because it is difficult to determine which form the ID is associated with. ■ The following items are *not* updated: ■ Some references in web services as mentioned earlier ■ References to the ID as data in a qualification (not a common occurrence) ■ References to the ID stored as data in any form, except as mentioned earlier These might include subsystems in applications that store qualifications or similar data that can contain references to field IDs. For example, BMC Remedy Approval Server, BMC Remedy Assignment Engine, AR System reports, BMC Service Level Agreement, and Crisis Response . Note: (*Oracle only*) Due to limitations in the Oracle 9i Release 2 and earlier databases, data for an Attachment field cannot be preserved when changing its ID. To preserve the attachment data, export the entry ID and attachment to a file, execute `archgid`, and reimport the data to the new ID by using the import-in-place option. IMPORTANT: Menus reference fields by using field IDs, not form IDs. Therefore the `archgid` utility does not know whether to update those references. You must check the updated form definition and if required, change the reference manually.

Object type	Corresponding updates
Group	■ The permission definitions of all workflow objects that reference this group are updated. ■ The following instances of the old ID are updated: ■ ID of the group in the Group form ■ ID in the Computed Group Definition field of the Group form ■ ID in the Group List field of the User form ■ ID in the Computed Group List field of the User form ■ ID in the mapped group fields of the Roles (role mapping) form ■ The references in active links and filters that use the special Run Process command `Application-Confirm-Group` are updated: ■ ID in Flashboards Data Source definitions ■ (optional) ID in field 112 in any form that contains field 112 ■ (optional) ID in any of the additional dynamic group fields ■ The following items are *not* updated: ■ Any references to the ID as data in a qualification statement that is used to search a field for a specific group ID ■ Role IDs

WARNING

■ During execution, the utility signals the AR System server to reset the cache so that it refers to the recently updated database structure and data definitions. If you access the system during this time, you might receive inconsistent responses. You should also restart AR System clients so that they refer to these changes.

■ If you execute this utility on a production server, the performance might be impacted. Depending on the operation requested, the impact could be severe. The changes being made to the data dictionary and database structures could lead to runtime errors. Users who access AR System while the server is being restarted to reflect the recent changes may receive inconsistent responses.

■ When identifying and updating field ID references, the utility does not consider the server name. If the form name matches, it is assumed to be the correct form regardless of the server on which it is located. If you have cross-server references and different forms with the same ID exist across different servers, this utility might update the wrong IDs and references.

■ For Web Services or Filter Set Fields that reference Web Services, the field ID change will work only for simple web services documents. If you have a complex web service document, changing the ID will not work correctly.

■ In general, after the change, the system will be up and fully functional with all references to the ID updated and ready to go. However, there may be situations where the ID is used in a way that we cannot correct. These cases are unusual, but they can occur. There is the possibility that some final, manual cleanup is required.

Syntax of archgid

This section describes the syntax of the `archgid` command, and uses the following conventions:

- Required parameters and their values are *not* enclosed in brackets.
- Optional parameters and their values are enclosed in square brackets.
- Braces indicate that you must specify only one of the enclosed paramaters or values at a time.

The syntax of `archgid` is as follows:

```
archgid -c commandCode [-i newID] [-q] [-o]
[{-s formName} |
{-s formName -f {fieldName | fieldID}} |
{-s formName -v {viewName | viewID}} |
{-g {groupName | groupID} [-y]}]
[{-F count~oldID~newID~formName~ | -F fileContainingIDsToChange}]
-u adminUser -p adminPassword [-a adminAuthenticationString]
-x serverName [-t TCPPort] [-r RPCNum]
```

Table F-1 lists the parameters of the `archgid` utility and the data types of the corresponding input values, and describes their usage.

Table F-1: archgid parameters (Sheet 1 of 3)

Parameter	Value type	Description

`c`	Integer	Command code—a number that indicates the type of object for which you want to change the ID. It also indicates the type of parameter-value pair that
		■ **1**—Form
		■ **2**—Field
		■ **3**—View
		■ **4**—Group
		Additionally, use the following command codes to update *only* field IDs or view IDs in bulk mode:
		■ **10002**—To change the specified *field* IDs (useful when changing a few field IDs). Specify multiple triplets of form, old field ID, and new field ID, separated by spaces and tilde characters (~) as follows:
		`count~oldID~newID~formName`
		The `~oldID~newID~formName` string is repeated for every form, for example:
		`oldID~newID~formName~`
		`archgid -c 10002 2~536871168~303549600~EMP:Contact Info ~536870913~303549300~EMP:Work Record -u Admin -p “” -x EmpMngtAppSvr`
		■ **10003**—To change *field* IDs listed in a file (useful when changing numerous field IDs). Specify the file name that contains triplets of form, old field ID, and new field ID, as follows:
		`archgid -c 10003 -F fileName -u Admin -p “” -x EmpMngtAppSvr`
		See description of the `-F` parameter in this table below.
		■ **10012**—To change the specified *view* IDs (useful when changing a few view IDs). The input format is the same as that for field IDs.
		■ **10013**—To change *view* IDs listed in a file (useful when changing numerous view IDs). The file format is the same as that for field IDs.
		Note: If you use a bulk mode command code (10002, 10003, 10012, 10013), the `-s`, `-f`, and `-n` parameters are ignored. If the utility encounters a problem (like an invalid form name or a nonexistent object), it issues a warning, skips the corresponding object, and continues processing.

`i`	Integer	New ID that you want to assign to the specified object.
`q`	None	Indicates whether to display the objects and the related database structures being processed by the utility: ■ (Default) In the absence of this parameter, the utility displays processing information on the screen. To record this information, redirect this output to a file. ■ To execute the utility in Silent mode, specify this parameter. No processing is displayed or written to a log.
`o`	None	Indicates whether to display the confirmation prompt before running the utility: ■ (Default) In the absence of this parameter, the utility displays a confirmation prompt before performing the updates. ■ To skip the confirmation prompt, specify this parameter.

Table F-1: archgid parameters (Sheet 2 of 3)

Parameter	Value type	Description
`s`	String[a]	Name of the form whose ID you want to change, or whose field ID or view ID you want to change.
`f`	Integer	Name or ID of the field whose ID you want to change.
`v`	Integer	Name or ID of the view whose ID you want to change.
`g`	Integer	Name or ID of the group whose ID you want to change.
`S`	String	(Optional) One or more field IDs separated by spaces, tabs, colons, semicolons, forward slash characters, or backslash characters that uniquely identify a form. The IDs must identify a *single* form. If multiple forms match, the utility returns an error and displays the matching forms. Note: This parameter is *ignored* if you specify the `-s` parameter.
`y`	None	Flag used when changing a group ID to indicate whether the utility should update related data fields. ■ (Default) In the absence of this parameter, the utility does update the related data fields. ■ To update the group ID in all row-level security data fields of all forms with the new ID, specify this parameter.

`F`	String	Name of the file that contains data for changing field IDs and view IDs.
		The data is in the form of triplets of old ID, new ID, and form name. The IDs and form names are separated by spaces or tabs, and the triplets are separated by new line characters. For example:
		`536871168 303549600 EMP:Contact Info`
		`536870913 303549300 EMP:Work Record`
		`536870913 303549200 EMP:Insurance Info`
		`536870915 303548700 EMP:DataLoadConsole`
		
		Note: This option should be used in conjunction with command codes 10002, 10003, 10012, and 10013.
		
		Note: If you use this parameter to specify multiple objects at the command line or in a file, the utility updates all of their IDs in a single run.
		
		Note: If you use this parameter, the utility ignores the `-s`, `-f`, and `-n` parameters.

`u`	String	Name of the administrator user who executes the utility.
`p`	String	Password of the specified administrator user.
`Z`	None	Indicates that the `archgid` utility should not use the special system call. This system call is used to mask the password, which in turn imposes the following restrictions on the utility: ■ It only accepts passwords up to eight characters. ■ It does not allow you to redirect input from a file. To overcome these restrictions, you can specify the `-Z` parameter. This prevents the use of the special system call. Note: If you specify this parameter, the password value that you enter is *not* masked. Use this parameter only if your password is longer than eight characters or if you want to run some reports with the redirected input .

Table F-1: archgid parameters (Sheet 3 of 3)

Parameter	Value type	Description
a	String	(*optional*) Authentication string for the specified administrator user. For Windows, where the NT domain is required for login, this is the domain name. Note: If omitted from the command line, you are prompted to provide this value only if the user name is required.
x	String	Name of the AR System server on which the object exists.
t	Integer	TCP port that the AR System server uses for communication. This parameter is optional *if* the AR System server is registered with the portmapper. Note: If omitted from the command line, you are prompted to provide this value only if the server name is required.
r	Integer	(*optional*) RPC number of the AR System server.

[a] If any parameter values of the String type contain spaces, enclose them in double quotation marks (for example, "`EMP:Contact Info Form`").

Examples

The following examples depict how to use the `archgid` utility to change IDs of various objects:

- Changing form ID

```
archgid -c 1 -i 5626368972 -s "EMP:Contact Info" -u Admin -p ""
-x EmpDataAppSvr -t 9999
```

- Changing view ID

```
archgid -c 3 -i 5402 -s "EMP:Contact Info" -v "Employee View"
-u Admin -p "" -x EmpDataAppSvr
```

- Changing field ID

```
archgid -c 2 -i 8282564391 -s "EMP:Contact Info" -f EmpName
-u Admin -p "" -x EmpDataAppSvr -t 9999 -r 16002
```

Using archgid in prompt-driven mode

In the prompt-driven mode, the `archgid` utility allows you to enter values for all the required parameters in a sequence. It displays the list of available input options on the screen and prompts you to specify a choice.

NOTE

When the list input options is significantly large, you might not be able to scroll up and view all the items.

The utility only prompts you to provide input for *required* parameters. To use the following *optional* parameters, you must specify them at the command line before pressing Enter:

- `-q`
- `-o`

- `-y`
- `-Z`

See "Syntax of archgid" on page 412.

To execute archgid in prompt-driven mode

1. Back up your database.

   The utility makes fundamental changes to the AR System database structure and data definitions. If you encounter any problems during the execution of `archgid`, you might need to restore the database to its earlier working condition.

2. At the command prompt, navigate to the folder where the `archgid` utility is located, type `archgid` and press Enter.

3. The utility prompts you sequentially for the user name, password, and authentication string.

   - The user must be an AR System administator user.

4. By default, the utility displays a list of servers that it retrieves from the `/etc/ar` file (UNIX) or the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\ARSystem\ARServer\CurrentVersion\ServerList` (Windows).

   - If multiple servers are listed in the file or the registry key, the utility displays a numbered list of their names, and prompts you for a choice as follows:

     ```
     1-serverName1  2-serverName2  3-serverName3  4-serverName4
     Enter id of server:
     ```

     Enter the number that corresponds to the server on which you want to change object IDs. You can only specify one server—the utility only executes on one server at a time.

   - If only one server is listed in the file or the registry key, the utility selects that server and displays a message as follows:

     ```
     Connecting to only server identified -- serverName
     ```

   To override this list of servers, specify the `-x` parameter followed by the appropriate server name.

5. If the utility prompted for a server name and you provided a valid one, it prompts for the TCP port to be used.

   - If required, specify the TCP port number. Otherwise press Enter to proceed.
   - If you specified the `-x` parameter and the server name as its value and if a TCP port is required, you must use the `-t` parameter to specify the TCP port value. The utility does not separately prompt for the TCP port.

6. The utility displays a numbered list of object types (whose ID you can change), and prompts you for a choice as follows:

   ```
   Type of ID to change (1 - form, 2 - field, 3 - VUI, 4 - group):
   ```

   Specify the number that corresponds to the object type for which you want to change the ID.

- If you specify `1` (form), the utility displays a numbered list of all the forms on a server and prompts you to choose one.
- If you specify `2` (field), the utility displays a numbered list of all the forms on a server, and prompts you to specify the one that contains the field whose ID you want to change. After you specify the form, it displays a numbered list of all the fields on that form and prompts you to choose one.
- If you specify `3` (view), the utility displays a numbered list of all the forms on a server, and prompts you to specify the one that contains the view whose ID you want to change. After you specify the form, it displays a numbered list of all the views on that form and prompts you to choose one.
- If you specify `4` (group), the utility displays a numbered list of all the groups on a server and prompts you to choose one. After you specify the group, it prompts you to choose whether to update entries that have row-level security fields. If you specify `Y`, the utility also updates the related data in the database.

7 After you specify the object type, the utility prompts you for the new ID.

- If the new ID is not unique, the utility reports that it conflicts with an existing ID and exits.
- If the new ID is unique, the utility accepts the input and proceeds further.

8 If you did not specify the `-o` paramater, the utility displays the choices you specified so far and prompts you for confirmation.

a Verify whether the displayed information is accurate.

b Confirm whether the utility should continue with the processing or not.

- If you specify `Y`, the utility performs the necessary updates.
- If you specify `N`, the operation is cancelled and no updates are made.

If you did not specify the `-q` paramater, information about the operations performed and the tables updated is written to the log file.

Using archgid in bulk mode

In bulk mode, the `archgid` utility updates the IDs of all the specified objects together in a single run. It consolidates performance and network-intensive updates (like those for active links, filters, and escalations) into a single batch. It retrieves the required definitions from the server, performs all updates for all forms, and returns the updated data in a single pass. To further speed processing, the utility updates the data dictionary for multiple items without resynchronizing the running cache.

▶ To execute archgid in bulk mode

1 Back up your database.

The utility makes fundamental changes to the AR System database structure and data definitions. If you encounter any problems during the execution of `archgid`, you might need to restore the database to its earlier working condition.

2 At the command prompt, navigate to the folder where the `archgid` utility is located.

3 Type `archgid -c` *`commandCode`*.

Only the following command codes indicate bulk mode:

- 10002
- 10012
- 10003
- 10013

TIP

You cannot use these command codes in the prompt-driven mode. You can provide the arguments associated with these codes only in the command-line mode.

Also specify all the optional parameters of your choice and press Enter.

NOTE

When using these codes, if you do not specify the *required* parameters (for example, server and user information), the utility prompts you for their values.

Index

A

access control, AR System 33
active link actions
  DDE 287, 291, 293
  OLE Automation 279, 280
active links, execution order 89
AIE. *See* BMC Atrium Integration Engine
aliases, plug-in 119
Analyzer
  custom rules 218
Apache AXIS, WSDL file 66
API
  BMC Remedy Developer Studio 220
applications
  licensing 323
  overview of licensing 325
AR System
  access control 33
  active links, described 32
  API, understanding 43
  architecture, described 24
  architecture, terminology 22
  C API
    clients 45
    program structure 44
    using for integration 46
  clients
    communications with server 30
    described 26
  communications
    between clients and server 30
    with database servers 30
  components 32
  data, accessing externally 198
  database server, described 29
  escalations, described 32
  external processes 260
  filters, described 32
  forms, described 32
  Java API
    cloning objects 53

AR System (continued)
  Java API (continued)
    exception handling 53
    installed files 50
    overview 50
    program model 52
  menus, described 32
  ODBC 202
  plug-ins 106
  security 33
  server, communications with clients 30
  server, described 28
  XML data 235
  XML objects 234
  XML, using with AR System API 235
AR System Database Connectivity. *See* ARDBC
AR System External Authentication. *See* AREA
AR System filter. *See* ARF
architecture
  AR System, described 24
  AR System, terminology 22
ARDBC LDAP
  Configuration form 137
  example
    attaching fields to a vendor form 365
    creating a vendor form 364
    defining a filter 367
    generating a DN 367
ARDBC LDAP plug-in. *See* LDAP
ARDBC plug-ins
  about 127
  calling AR System API 129
  creating vendor forms for 130
  functions 128
AREA
  authenticating unregistered users 155
  authentication chaining mode, and 156, 157
  configuring 155
  cross referencing blank passwords 156
  hub, setting up 164
  overview 154
  plug-ins

AREA (continued)
  plug-ins (continued)
    about 124
    functions 126
    sample implementations 127
  when to use 155
ARF plug-ins
  about 132
  functions 133
armonitor.conf (armonitor.cfg) file, SNMP and 318
arplugin server 120
arsnmp file 312
arsystem shell script 318, 319
attributes
  missing XML 354
  object in directory services 143
  trust 403
authentication chaining mode
  effect on authentication processing 157
  specifying 156
authentication processing
  authenticate unregistered users option 155
  authentication chaining mode, specifying 156
  configuring 155
  cross reference blank passwords option 156
automation servers (OLE) 281

B

basic web services 72
Bind User 147
BMC Atrium Integration Engine 221
BMC Remedy Data Import
  commands and options 245
  examples 256
  without mapping 255
BMC Remedy Developer Studio
  custom Analyzer rules 218
  custom context menu items for object lists 218
  custom context menu items for servers 218
  custom server object types 217
  customizing 215
  described 26
  Eclipse plug-ins 216
  exporting definitions 235
  exporting objects (CLI) 239
  extending 215
  extension points 216, 219
  importing definitions 235
  importing objects (CLI) 239
  Java API 220
  plug-in directory 220
BMC Remedy Developer Studio (continued)
  plug-in prerequisites 219
  plug-ins 216
  project dependencies 219
BMC Remedy Mid Tier, described 28
BMC Remedy SNMP Agent. *See* SNMP
BMC Remedy User
  Crystal Reports 205
  described 26
  macros, third-party applications 296
  ODBC driver installed 202
  path and DDE server name 296
BMC Software, contacting 2

C

C API
  clients 45
  overview 42
  program structure 44
  understanding 43
  using for integration 46
cert7.db file 401
cert8.db file 401, 402
certificate databases
  about 402
  adding certificates to 403
  cert7.db and cert8.db files 401, 402
  configuring
    ARDBC LDAP plug-ins 138
    AREA LDAP plug-ins 147
  creating 402
  listing certificates 404
  trust attributes 403
certificates
  adding to certificate databases 403
  listing in certificate databases 404
  viewing information about 404
certutil utility 402
choice element, web services 103
com.bmc.arsys.studio.analyzer.core 219
com.bmc.arsys.studio.analyzer.core.analyzerRules 2
  20
com.bmc.arsys.studio.api 219
com.bmc.arsys.studio.commonui 219
com.bmc.arsys.studio.commonui.genericaction 220
com.bmc.arsys.studio.commonui.typeaction 220
com.bmc.arsys.studio.commonui.typeinformation 2
  20
com.bmc.arsys.studio.model 219, 220
com.bmc.arsys.studio.model.modelprovider 219
com.bmc.arsys.studio.model.modeltype 219

com.bmc.arsys.studio.ui 219
command, DDE 292
command-line utilities. *See* utilities
complex documents
  filter flow 345
  hierarchical, web services 342
  publishing (web services) 381, 394
configuration files
  arsnmpd 312
  snmpd 317
configuring
  applications for licensing 326
  AR System with internet access through proxy server 67
  Crystal Reports 205
  LDAP plug-in 137
  SNMP 311
  traps for SNMP 316
consuming
  complex documents, web services 394
  simple flat documents, web services 377
create operation type (web services) 332
creating
  objects (LDAP) 142
  plug-ins 110
  vendor forms 140
  vendor forms for ARDBC plug-ins 130
  web services 61, 72
Crystal Reports
  date/time strings 210
  join forms 210
  limitations 210
  login, AR System 205
  report fields in 208
  setting up 205
  sorting in lists 210
  using 205
customer support 3
customizing
  BMC Remedy Developer Studio 215

D

data
  importing 245
  organizing, in directory services 140
  types, web services 360
data visualization field
  creating 174
  finding examples 175
  Java classes 171
  native libraries 172
data visualization field (continued)
  overview 168
  registering 178
  right-to-left format 182
  services and BMC Remedy Mid Tier 168
  services and clients 170
  storing shared library files 172
databases
  certificate 402
DDE
  action 291
  active link values, setting 293
  active links 287
  BMC Remedy User macros 296
  DoExecMacro topic 297
  examples 297, 300, 303
  executing macros 287
  fields, setting values 293
  item name 286
  keyword 293
  macros, samples 300
  Microsoft Excel integration 300–303
  Microsoft Word integration 303–305
  overview 286
  request operation result syntax 293
  RunMacro function 297
  server name for BMC Remedy User 296
  service name 286
  time-out settings 290
  topic name 286
  win.ini configuration 296
DefinitionExport options 236
DefinitionImport options 236
dependencies, BMC Remedy Developer Studio project 219
deployable applications
  configuring to license 326
  fixed licenses 324
  floating licenses 324
  license options 324
  licenses on server, applying 327
  licenses to license pools, applying 329
  licenses to users, applying 328
  licensing (overview) 323
  licensing of forms not reversible 326
  read licenses 324
Developer Studio. *See* BMC Remedy Developer Studio
directives, SNMP
  community-based 314
  user-based 314
directory services

directory services (continued)
  defined 140
  distinguished name attribute 141
  mapping data 140
  object attributes in 143
  objects in 140
  organizing data 140
Distinguished Name
  AREA LDAP configuration 147
  defined 141
  defining filters for 367
  described 143
documentation, AR System 13
documents
  AR System 303
  complex hierarchical, web services 342
  complex, filter flow 345
  simple, web services 340
  web services and style 64
DoExecMacro DDE topic 297
dynamic data exchange. *See* DDE

E

Eclipse plug-ins 216
escalations, execution order 89
examples
  BMC Remedy Data Import 256
  DDE
    integration with MS Excel 300
    integration with MS Word 303
    program and buffer 297
    requests, assigning values from 293
  export 239
  import 239
  OLE automation active link action 283
  runmacro 242
  web services
    complex document, consuming 394
    complex document, publishing 381
    simple flat document, consuming 377
    simple flat document, creating 372
execution order, active links and filters 89
export
  command-line options 236
  examples 239
  objects (CLI) 239
extending
  BMC Remedy Developer Studio 215
extension points, BMC Remedy Developer Studio 216, 219
extensions, BMC Remedy Developer Studio 216
external processes, running from AR System 260

F

fields
  See also individual database by name
  database name, identifying 208
  inetorgperson form, summary 368
  modifying in view and vendor forms 194
  values, setting with DDE request results 293
filters
  Distinguished Name, defining for 367
  execution order 89
  flow for complex documents 345
fixed licenses, deployable applications 324
flat mapping, XML 357
floating licenses, deployable applications 324
forms
  ARDBC LDAP Configuration 137
  ARDBC plug-in vendor 130
  directory services, building 140
  licensing in deployable application not reversible 326
  mapping to collection of objects (LDAP) 140
  vendor, creating 140
functions
  ARDBC plug-in 128
  AREA plug-in 126
  ARF plug-in 133

G

get operation type (web services) 333
get operation type, complex documents 345
global information, protecting plug-in 111
guidelines, import and export utilities 236

H

HARDWARE keyword 280, 291
holiday time. *See* business time
https, accessing WSDL or web services over 68

I

import
  command-line options 236
  data 245
  examples 239
  external XML schema 358
  objects (CLI) 239

import (continued)
- with mapping 255

importing
- multithreaded environments 253

inetorgperson
- summary of fields in example 368
- vendor form, example 364

input values, plug-in 111
installing plug-in components 108
integration
- areas for 18
- asynchronous 18
- at the AR System client 36
- at the AR System server 36
- at the database 36
- benefits 18
- C API, overview 42
- defined 17
- methods 38
- Microsoft Excel with AR System 300
- Microsoft Word with AR System 303
- points, choosing 38
- real-time 18
- with AR System 19

integration point
- AR System client 36
- AR System server 36
- C API 42
- choosing 38
- database 36
- multiplatform issues 37

integration system vendors (ISVs), licensing applications 323
internet access through proxy server, configuring 67
ISVs. *See* integration system vendors
item name (DDE) 286

J

Java API
- BMC Remedy Developer Studio 220
- exception handling 53
- installed files 50
- objects, cloning 53
- online documentation 53
- overview 50
- program model 52

Java data import utility 243
Java plug-in 112
- API, online documentation for 112

JavaScript
- limitations 267

JavaScript (continued)
- Run Process action, running from 266

join forms
- Crystal Reports and 210
- web services 350

K

keywords
- DDE 293
- HARDWARE 280, 291

L

LDAP
- ARDBC LDAP configuration form 137
- AREA plug-in
  - configuring 152
  - configuring group search 150
  - configuring groups 152
  - mapping groups 151
- certificate databases and 402
- configuring certificate databases for ARDBC plug-ins 138
- configuring certificate databases for AREA plug-ins 147
- defined 136
- distinguished name 147
- mapping data 140
- object creation 142
- overview of plug-in 136
- performance tips 144
- plug-in, configuring 137
- Request ID field limitations 142
- requirements of plug-in 136
- troubleshooting tips 144
- URL standard 140

libraries, native 172
library files, shared 172
licenses
- *See also* deployable applications
- application, managing 328
- deployable applications overview 324
- options for deployable applications 324
- overview of application 325

limitations
- Crystal Reports 210
- Request ID field with LDAP 142
- web services 101
- WSDL, for consumption 94

line items, web services 344
list of web services, viewing 80

logging in, Crystal Reports and 205

M

macros
- DDE programming 287, 296
- MS Office applications, examples 300
- runmacro utility 240
- third-party applications 296

mapping data
- BMC Remedy Data Import utility 255
- LDAP 140

memory, managing plug-in 111
methods (OLE)
- adding 281
- deleting 281

methods of integration 38
Microsoft
- .NET, WSDL file 66
- *See also* Internet Explorer
- Access, integration with AR System 211
- Excel, integration with AR System 300
- Word, integration with AR System 303

modifying fields in view and vendor forms 194
mutlithreaded environments
- importing 253

N

naming plug-ins 117
native libraries 172
nillable attributes, web services 354

O

object classes
- defined 142
- objectclass attribute 142

objects
- creation, supporting 142
- directory service 140
- identifying uniquely 141
- properties, simple XML editing 352

ODBC
- clients, compatibility with 205
- Crystal Reports and 205–210
- data sources, adding multiple 202
- driver installed with BMC Remedy User and mid tier 202
- Microsoft Access and 211
- Microsoft Excel and 212

ODBC (continued)
- overview 202

OLE
- active links 280
- Automation active link action
  - example 283
- automation active link action 279
- methods, adding 281

operations
- types, web services 332

options
- BMC Remedy Data Import utility 245
- DefinitionExport 236
- DefinitionImport 236
- runmacro 241

P

path (DDE) 292
performance tips for LDAP 144
plug-in API
- installing 108
- Java
  - abstract classes 112
  - interfaces 112
  - online documentation 112

plug-in server
- configuring 114
- installing 108
- running 120

plug-ins
- *See also* ARDBC plug-ins; AREA plug-ins; ARF plug-ins; plug-in server
- about 106
- aliases 119
- AR Filter 107
- ARDBC 106, 127
- ARDBC LDAP 137
- AREA 106, 124
- AREA LDAP 145
  - configuring 152
  - configuring group search 150
  - configuring groups 152
  - mapping groups 151
- ARF 132
- arplugin server 120
- BMC Remedy Developer Studio 216
- BMC Remedy Developer Studio prerequisites 219
- certificate databases and 402
- conventions 111
- creating 110

plug-ins (continued)
  Eclipse 216
  global information 111
  input values 111
  installing components 108
  Java 112
  memory 111
  multithread safety 111
  naming 117
  port numbers 118
  return values 111
  service 114, 120
points, BMC Remedy Developer Studio extension 216, 219
port numbers, plug-in 118
predefined web services 64
product support 3
project dependencies, BMC Remedy Developer Studio 219
proxy server
  configuring AR System with internet access through 67
  configuring web services with 67
  Java plug-in server and 67
publishing
  complex document, web services 381
  simple flat document, web services 372
  web services 72

R

read licenses, deployable applications 324
Remote Procedure Call. *See* RPC message style
reporting
  configuring for DDE 293
  logging in from Crystal Reports 205
Request ID field
  identifying objects uniquely 141
  LDAP limitations 142
retrieving data from another application 264
return values, plug-in 111
right-to-left format 182
RPC message style 64
rules, custom Analyzer 218
Run Process action
  client and server processes, and 260
  retrieving data from another application 264
  running a process on the web 266
  running JavaScript on a browser 266
  using to start an external application 260
runmacro
  commands and options 240
runmacro (continued)
  DDE function 297
  examples 242

S

schema
  constructs not supported, XML 103
Secure Sockets Layer. *See* SSL
security, AR System 33
servers
  application licenses, applying 327
  automation (OLE) 281
  DDE name 296
  plug-in 120
service name (DDE) 286
service operation type (web services) 334
service, plug-in 120
set operation type
  complex documents 344
  web services 332
set operation type (web services) 332
shared library files 172
simple documents
  flat, consuming (web services) 377
  flat, publishing (web services) 372
  web services 340
Simple Network Management Protocol. *See* SNMP
simple XML editing
  null, empty, or missing values, handling 354
  object properties 352
SNMP
  access control 313
  AR System processes monitored 309
  armonitor configuration file 316, 318
  arsnmp file 312
  community-based directives 314
  configuration 311
  configuration files 311
  directives 313, 314
  overview 308
  Remedy MIB 310
  Remedy SNMP Agent functions 309
  snmpd file 317
  starting Remedy SNMP Agent 318
  stopping Remedy SNMP Agent 319
  system information 313
  trap configuration 316
  traps 310
  troubleshooting 319
  user-based directives 314
  versions supported 308

SOAP
  encoding rules for web services 64
  headers and authentication 95
  web services 62
spreadsheets, in AR System 300
SSL
  certificate databases and 402
support, customer 3
SystemWebService.def file 64

T

technical support 3
third-party applications and macros 296
time-out settings, DDE 290
topic name (DDE) 286
troubleshooting LDAP 144
trust attributes 403
type library information (OLE) 281

U

URLs
  LDAP 140
  protected permissions for WSDL file 79
  WSDL (Apache AXIS) 66
  WSDL (Microsoft.NET) 66
  WSDL file 79
  XSD file 358
users
  application licenses, applying 328, 329
utilities
  BMC Remedy Data Import 243
  certutil 402
  DefinitionExport 235
  DefinitionImport 235
  runmacro 240, 241, 242

V

values, handling null, empty, or missing 354
vendor forms
  about 184
  attaching fields, example 365
  creating for
    ARDBC plug-ins 130
    collections of LDAP objects 140, 364
  example 364
  inetorgperson example 364
  troubleshooting 144
  using 184
view forms
  data types supported 190
  modifying fields 194
viewing, list of available web services 80

W

warnings
  licensing of forms in application not reversible 326
web services
  accessing, over https 68
  advanced XML editing 358
  authentication 95
  basic 65, 72
  basic, creating 72
  choice element 103
  clients, writing 66
  complex documents 342
  Configuration Tool for Remedy mid tier 96
  create operation 332
  creating 72
  custom, creating 75
  data types 360
  examples 372
  external XML schema, importing 358
  fetch record
    setting maximum 337
    setting starting record 337
  get operation 333, 345
  internet access through a proxy server 67
  join forms 350
  limitations 101
  line items 344
  list, viewing 80
  mapping 340, 345
  nillable attributes 354
  object properties 352
  operations 77, 332
  predefined 64
  proxy server 67
  publishing 72
  service operation 334
  set operation 332, 344
  simple documents 340
  simple XML editing 352
  SOAP protocol 62
  XML editing 351, 358
  XPATH function 335
workflow actions
  DDE 291
  OLE Automation 279

WSDL
  accessing, over https 68
  file 75, 78, 79
  limitations for consumption 94
  protected permissions in URL 79
  public permissions in URL 79
  URL to WSDL file 79
  web services implementation 62
  WSDL file (Apache AXIS) 66
  WSDL file (Microsoft.NET) 66

X

XML
  advanced editing 358
  AR System API and 235
  data and AR System 235
  flat mapping 357
  missing attributes 354
  objects and AR System 234
  schema constructs not supported 103
  simple editing 352
  URL with XML schema 358
  web services, editing 351
XPATH function
  web services 335, 337