

BMC® Remedy® Action Request System® 7.0

Integrating with Plug-ins and Third-Party Products



March 2006
Part No: 58467

Copyright 1991–2006 BMC Software, Inc. All rights reserved.

BMC, the BMC logo, all other BMC product or service names, BMC Software, the BMC Software logos, and all other BMC Software product or service names, are registered trademarks or trademarks of BMC Software, Inc. All other trademarks belong to their respective companies.

BMC Software, Inc., considers information included in this documentation to be proprietary and confidential. Your use of this information is subject to the terms and conditions of the applicable end user license agreement or nondisclosure agreement for the product and the proprietary and restricted rights notices included in this documentation.

For license information about the OpenSource files used in the licensed program, please read *OpenSourceLicenses.pdf*. This file is in the \DOC folder of the distribution CD-ROM and in the documentation download portion of the product download page.

Restricted Rights Legend

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC Software, Inc., 2101 CityWest Blvd., Houston, TX 77042-2827, USA. Any contract notices should be sent to this address.

Contacting Us

If you need technical support for this product, contact Customer Support by email at support@remedy.com. If you have comments or suggestions about this documentation, contact Information Development by email at doc_feedback@bmc.com.

This edition applies to version 7.0 of the licensed program.

Contents

Preface	13
Audience	13
AR System documents	14
Learn about the AR System Developer Community	16
Why should you participate in the Developer Community?	16
How do you access the Developer Community?	16
Chapter 1 What does “integration” mean?	17
Benefits	18
Areas for integration	18
Real-time versus asynchronous integration	19
Integrating with AR System	19
Chapter 2 Architectural overview of AR System	21
Terminology	22
Form	22
Request	22
Workflow	22
Application	22
Multi-tier architecture	23
AR System clients	25
BMC Remedy User	25
Web and wireless clients	25
BMC Remedy Administrator	25

BMC Remedy Import	25
BMC Remedy Alert	26
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.	31
AR System components	32
Security and access control	35
 Chapter 3 Integration considerations	37
Where to integrate.	38
AR System client	38
AR System server	39
Database integration	39
Multi-platform issues	40
Choosing an implementation method	40
Integration technologies	41
 Chapter 4 AR System C API	45
Overview	46
Understanding the AR System API	47
Program structure	49
Multithreaded API clients	49
Using the API for integration	51
Example: Network management platform integration accessories.	52
Issues and considerations.	53
 Chapter 5 Java API	55
Overview	56
Programming model.	56
Object factories	57
Criteria objects	58
Criteria list objects.	58
Cloning objects	58

Exception handling	58
Java API program structure.	59
Naming conventions: how C structures and functions correspond to Java API objects	68
Exceptions to naming conventions	69
Installation and deployment	69
Contents of the AR System java object installation	70
Windows environment setup	71
UNIX environment setup	72
Issues and considerations.	73
Chapter 6 Web services	75
Overview of web services	76
Web service architecture	76
Writing web service clients	81
Setting up your environment for web services.	81
Java runtime environment considerations	82
Configuring AR System with Internet access through a proxy server	82
Creating a web service	87
Creating a basic web service	87
Creating a custom web service	91
Saving your web service	98
Viewing a list of web services	98
Publishing a web service	99
SOAP headers and authentication	103
Consuming web services	105
Flow for consuming a web service	106
Creating the set fields filter to import an external web service	106
Consuming a web service published on the same AR System server	109
Web services limitations	110
XML schema limitations	110
XML schema constructs not supported in AR System.	112
WSDL limitations for consumption	113

Chapter 7	Plug-ins	115
	Overview of AR System plug-ins.	116
	Creating plug-ins	118
	Common plug-in API functions.	120
	Running the arplugin server	121
	Accessing the plug-ins	122
	Issues and considerations	122
	AR system external authentication (AREA) Plug-Ins.	122
	AREA plug-in specific API functions.	124
	Issues and considerations	125
	Filter Plug-Ins.	126
	Filter Plug-In specific API function	127
	Issues and considerations	127
	ARDBC Plug-Ins (AR System database connectivity) and vendor forms	127
	ARDBC Plug-In specific API functions.	128
	Calling AR system API from ARDBC plug-in	132
	Creating a vendor form using an ARDBC Plug-In	133
	Issues and considerations	136
	Logging options	136
	Plug-in aliases.	137
	Defining plug-in aliases	137
	Examples of plug-in aliases	137
	Plug-in port numbers	138
Chapter 8	LDAP plug-ins	139
	Overview of LDAP and AR System.	140
	ARDBC LDAP plug-in	140
	Requirements	140
	Configuring the ARDBC LDAP plug-in	141
	Building AR system forms for directory services	145
	ARDBC LDAP runtime performance tips.	151
	AREA LDAP plug-in	152
	Configuring the AREA LDAP plug-in	152
	Configuring your AR system server to use the AREA LDAP plug-in.	159
	Configuring AREA LDAP group search	160

Mapping LDAP groups to AR System groups	160
Configuring external authentication processing	162
Chapter 9 AR System external authentication (AREA)	163
Overview of AR system external authentication (AREA)	164
About the AREA LDAP plug-in	164
Specifying AREA plug-in server configuration settings	165
Configuring authentication processing	165
Specifying when to use internal and external authentication	165
Specifying authentication chaining mode	167
Determining AREA behavior	169
Setting up the AREA hub	174
Chapter 10 Data visualization fields	177
Overview	178
Services provided to the data visualization modules on BMC Remedy Mid Tier	178
Services provided on clients	180
Using Java classes	181
Creating data visualization fields	183
Creating data visualization modules on the mid tier	183
Registering data visualization modules	187
Chapter 11 Vendor forms	191
About vendor forms	192
Creating vendor forms	193
Chapter 12 View forms	197
Overview	198
Setting up a remote database for view forms	202
Field properties for fields on view forms	203
Modifying fields on view forms	204
Issues and considerations	204
Chapter 13 SQL database access	205
Accessing AR System data externally	206

Pushing data from AR System with SQL	206
Pulling data into AR System with SQL	207
Issues and considerations.	208
Chapter 14 ODBC database access	209
Overview	210
Creating multiple data sources	211
Compatibility with ODBC clients	214
Using Crystal Reports with AR System	214
Using field labels or database names in Crystal Reports	218
Crystal Report report options considerations	219
Selecting report fields in Crystal Reports	219
Using Crystal Reports with join forms	220
Limitations when using Crystal Reports	220
Using Microsoft Access with AR System	221
Using Microsoft Excel with AR System	222
Issues and considerations.	223
Chapter 15 Enterprise integration engine (EIE)	225
Chapter 16 Command-line interface (CLI)	227
Overview	228
Using the BMC Remedy Administrator CLI	229
Guidelines for using BMC Remedy Administrator CLI	229
BMC Remedy Administrator commands and options	230
BMC Remedy Administrator CLI examples	234
Using the BMC Remedy User CLI	236
Using the runmacro CLI	237
runmacro CLI example	240
Using the BMC Remedy Import CLI	240
Running arimportcmd on a UNIX machine.	241
Importing with mapping.	241
Importing without mapping	244
BMC Remedy Import CLI examples	246

Chapter 17	XML import and export	249
	AR System objects in XML	250
	AR System data in XML	250
	Using XML with the AR System API	251
Chapter 18	Running external processes (Run Process)	253
	Overview	254
	Client and server processes	254
	Implementing the Run Process action to execute another application	255
	Example: open a reference document from an active link button	258
	Example: call a pager application from a filter	259
	Implementing the Run Process action to retrieve data from another application	260
	Run a process on the web	263
	Limitations in using JavaScript	264
	Issues and considerations	264
Chapter 19	OLE automation	267
	OLE overview	268
	AR System and OLE automation	269
	Active links and OLE automation	270
	Using the GUID	272
	The OLE automation active link interface	272
	Reading the method tree	274
	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	280
	Issues and considerations	285
Chapter 20	Dynamic data exchange (DDE)	287
	Overview	288
	DDE parameters used by AR System	288
	Methods of integration	289

Configuring your system to use DDE with AR System	290
Working with your dde.ini file	290
DDE time-out settings.	293
Using active links with DDE	294
Using the DDE active link action	294
Using the DDE active link keyword	298
Using BMC Remedy User reporting with DDE	299
Configuring BMC Remedy User to pass report data	299
Creating a report for DDE export	300
Triggering AR System using a DDE execute from an external application.	302
Using AR System with DDE, third-party applications and macros	303
DDE server name and BMC Remedy User path	303
Supported DDE topic and function	304
Example program and buffer	304
Examples	307
Integrating with Microsoft Excel	307
Integrating with Microsoft Word	312
Using DDE to pass data to Excel for graphing	316
Integrating with CTI middleware application	317
Issues and considerations.	318
 Chapter 21 Simple network management protocol	319
Overview	320
BMC Remedy SNMP agent functions	321
Monitoring AR system.	321
Sending traps	323
SNMP configuration.	324
The arsnmpd configuration file	325
System information	326
Access control information.	326
Location of the armonitor configuration file	330
The snmpd configuration file	331
The armonitor file	332
Starting the Remedy SNMP Agent	333
Stopping the Remedy SNMP Agent	333
Troubleshooting	334

Chapter 22	Using source control	337
	Integrating source control with AR System	338
	Enforced and advisory modes.	339
	Setting up source control with AR System	341
	Integrating AR System source control with PVCS	344
	Integrating AR system source control with ClearCase.	345
	AR System source control options	347
	Adding AR System objects to source control	349
	Exporting AR System objects into source control	350
	Importing definitions from source control	352
	Removing objects in source control	355
	Checking objects in and out of source control.	356
	Checking out AR System objects	356
	Undoing a check-out of AR System objects	357
	Checking in AR System objects	357
	Viewing history in source control	358
	Refreshing the status history in source control.	359
	Getting the latest version of AR System objects	359
	Displaying user information in source control	360
	Running the source control client executable	361
Chapter 23	Making applications licensable for integration system vendors .	363
	Application licensing options	364
	Application licensing overview	365
	Configuring your applications to make them licensable	367
	Applying the application license on your server	369
	Assigning application licenses to users	371
Appendix A	Web service operation types	375
	Create operation type	376
	Set operation type	377
	Get operation type.	378
	XPATH function	379
	Setting the starting record and setting the maximum limit	381

Appendix B	Mapping web service data.	383
	Mapping to simple and complex documents	384
	Simple documents	384
	Complex hierarchical documents	386
	Mapping to complex documents	390
	Using join forms in web services	395
	XML editing	397
	Simple XML editing.	397
	Object properties	398
	Handling null, empty, and missing values.	399
	Advanced XML editing	404
	Data types	408
Appendix C	Web service examples.	411
	Example 1: publishing a simple flat document.	412
	Example 2: consuming a simple flat document	417
	Example 3: publishing a complex document	423
	Example 4: consuming a complex document	440
Appendix D	Sample exercise: using OLE automation	449
	Procedure 1—Creating the form	450
	Procedure 2—Defining the active link	451
	Procedure 3—Defining action 1.	452
	Procedure 4—Defining action 2.	454
	Procedure 5—Defining action 3.	456
	Procedure 6—Defining action 4.	458
	Procedure 7—Defining action 5.	459
	Procedure 8—Testing the active link.	462
Appendix E	ARDBC LDAP example: accessing inetorgperson data	465
	Creating the inetorgperson vendor form	466
	Attaching fields to represent inetorgperson data.	469
	Defining a filter to generate a DN	472
	Summary of fields.	475
Index.		477

Preface

Important: The compatibility information listed in the product documentation is subject to change. See the compatibility matrix at <http://supportweb.remedy.com> 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 Administrator and BMC Remedy User. In addition, it would be 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 products.

Unless otherwise noted, online documentation in Adobe Acrobat (PDF) format is available on AR System product installation CDs, on the Customer Support site (supportweb.remedy.com), or both.

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

Title	Description	Audience
<i>Concepts</i>	Overview of AR System architecture and features with in-depth examples; includes information about other AR System products as well as a comprehensive glossary for the entire AR System documentation set.	Everyone
<i>Installing</i>	Procedures for installing AR System.	Administrators
<i>Getting Started</i>	Introduces topics that are usually only learned when first starting to use the system, including logging in, searching for objects, and so on.	Everyone
<i>Form and Application Objects</i>	Describes components necessary to build applications in AR System, including applications, fields, forms, and views.	Developers
<i>Workflow Objects</i>	Contains all of the workflow information.	Developers
<i>Configuring</i>	Contains information about configuring AR System servers and clients, localizing, importing and exporting data, and archiving data.	Administrators
<i>Installing and Administering BMC Remedy Mid Tier</i>	Contains information about the mid tier, including mid tier installation and configuration, and web server configuration.	Administrators
<i>Integrating with Plug-ins and Third-Party Products</i>	Discusses integrating AR System with external systems using plug-ins and other products, including LDAP, OLE, and ARDBC.	Administrators /Developers
<i>Optimizing and Troubleshooting</i>	Server administration topics and technical essays related to monitoring and maintaining AR System for the purpose of optimizing performance and troubleshooting problems.	Administrators

Title	Description	Audience
<i>Database Reference</i>	Database administration topics and rules related to how AR System interacts with specific databases; includes an overview of the data dictionary tables.	Administrators
<i>Administering BMC Remedy DSO</i>	Server administration and procedures for implementing a distributed AR System server environment with the BMC Remedy Distributed Server Option (DSO).	Administrators
<i>Administering BMC Remedy Flashboards</i>	Flashboards administration and procedures for creating and modifying flashboards and flashboards components to display and monitor AR System information.	Administrators /Programmers
<i>C API Reference</i>	Information about AR System data structures, C API function calls, and OLE support.	Administrators /Programmers
<i>C API Quick Reference</i>	Quick reference to C API function calls.	Administrators /Programmers
<i>Java API</i> *	Information about Java classes, methods, and variables that integrate with AR System.	Administrators /Programmers
<i>Administering BMC Remedy Email Engine</i>	Procedures for installing, configuring, and using the BMC Remedy Email Engine.	Administrators
<i>Error Messages</i>	List and expanded descriptions of AR System error messages.	Administrators /Programmers
<i>Master Index</i>	Combined index of all books.	Everyone
<i>Release Notes</i>	Information about new features list, compatibility lists, international issues, and open and fixed issues.	Everyone
<i>BMC Remedy User Help</i>	Procedures for using BMC Remedy User.	Everyone
<i>BMC Remedy Import Help</i>	Procedures for using BMC Remedy Import.	Administrators
<i>BMC Remedy Administrator Help</i>	Procedures for creating and modifying an AR System application for tracking data and processes.	Administrators
<i>BMC Remedy Alert Help</i>	Procedures for using BMC Remedy Alert.	Everyone
<i>BMC Remedy Mid Tier Configuration Tool Help</i>	Procedures for configuring the BMC Remedy Mid Tier.	Administrators

*: A JAR file containing the Java API documentation is installed with the AR System server. Typically, it is stored in `C:\Program Files\AR System\Arserver\Api\doc\ardoc70.jar` on Windows and `/usr/ar/<server_name>/api/doc/ardoc70.jar` on UNIX.

Learn about the AR System Developer Community

If you are interested in learning more about AR System, looking for an opportunity to collaborate with fellow AR System developers, and searching for additional resources that can benefit your AR System solution, then this online global community sponsored by BMC Remedy is for you.

In the Developer Community, you will find collaboration tools, product information, resource links, user group information, and be able to provide BMC Remedy with feedback.

The Developer Community offers the following tools and information:

- Community message board
- Community Downloads
- AR System Tips & Tricks
- Community recommended resources
- Product information
- User Experience Design tips

Why should you participate in the Developer Community?

You can benefit from participating in the Developer Community for the following reasons:

- The community is a direct result of AR System developer feedback.
- BMC Remedy provides unsupported applications and utilities by way of Community Downloads, an AR System application.
- BMC Remedy posts the latest AR System product information in the Developer Community to keep you up to date.
- It is an opportunity to directly impact product direction through online and email surveys.
- It's free!

How do you access the Developer Community?

Go to supportweb.remedy.com, and click the Developer Community link.

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 multi-platform, multi-database architecture and highly customizable user interface allow 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 19)
- Integrating with AR System (page 19)

Benefits

The primary intent of business software is to allow users to do their jobs more quickly using fewer resources. Using two products separately is usually less efficient than using them in an integrated fashion. For example, a user might have to type in the same information into two different applications, which often results in errors. Or, the telephone number of an incoming call might be entered manually 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 automatically launches another “in context,” so that the second application “knows” everything that has been entered into the first application, and the user is immediately focused at the part of the second application that continues the process. Or, the second application does its job automatically in the background based on directions from the first application, 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 about the AR System API, see the *C API Reference* guide.

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. The BMC Remedy User and BMC Remedy Administrator tools are built on the AR System C API. The BMC Remedy Mid Tier is built on the AR System Java API.

AR System has a multi-tier 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 the AR System.

The following topics are provided:

- Terminology (page 22)
- Multi-tier architecture (page 23)
- AR System clients (page 25)
- 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 31)
- AR System components (page 32)
- Security and access control (page 35)

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.

A form relates to a table in a database, and the fields on a form relate 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 then 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 to AR System 4.0, many API calls still use that term.)

Workflow

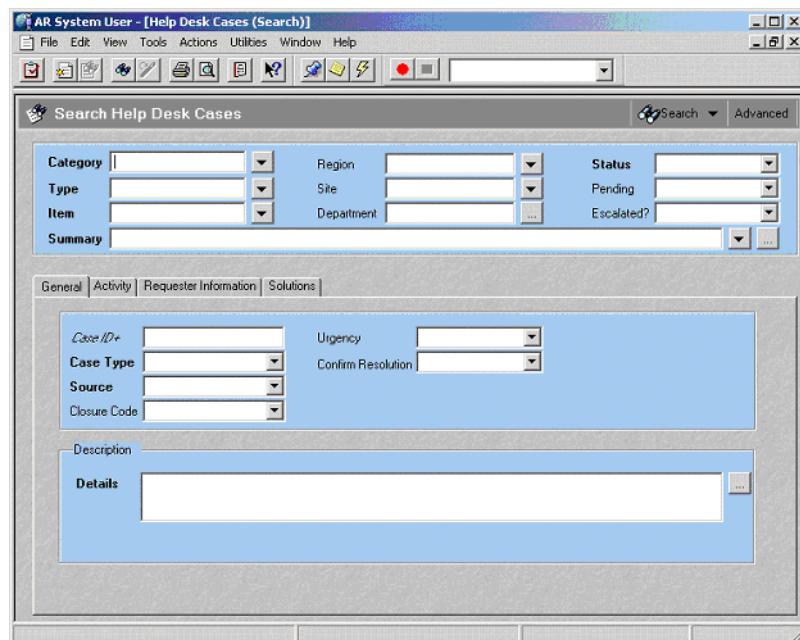
When data is written into a database or into a form on the client, it sits there until someone or some application decides to change it. *Workflow* provides a means to take action based on this data, and the actions can result in changes to the data. Workflow is composed of descriptions (sometimes called “definitions” or “rules”) that specify the actions to be taken 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 has been 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, or 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.

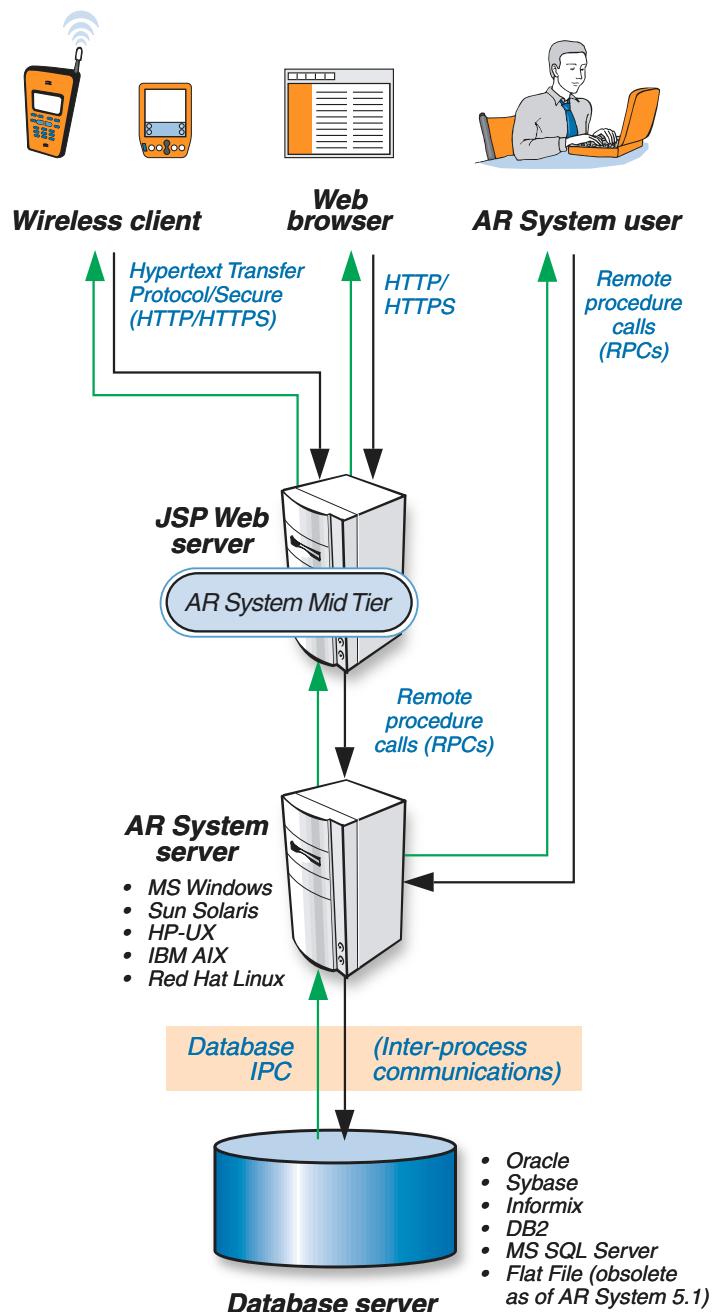
Figure 2-1: Sample BMC Remedy form



Multi-tier architecture

AR System is a multi-tier client/server architecture, as shown in Figure 2-2. AR System clients provide the user interface. The AR System Web server makes the user interface available from browsers. The AR System server implements the workflow functions, access control, and flow of data in to and out of the database. The database server acts as a data storage and retrieval engine. (The compatibility matrixes available at <http://supportweb.remedy.com/CompatibilityMatrix/index.html> detail which versions of the different platforms are supported.)

Figure 2-2: AR System Multi-tier Architecture



AR System clients

AR System clients are available for a number of operating system environments, as listed in Figure 2-2. 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:

BMC Remedy User

Provides the day-to-day user interface to AR System applications. It is used to submit and modify new requests, to search for information about requests, and to generate reports. BMC Remedy User tool is available for Windows platforms. An example of the interface is shown in Figure 2-1.

Web and wireless clients

Provide a web-based interface to AR System applications. Users can access these applications with browsers. The web pages are written in JSP and rendered in JavaScript and HTML.

The Remedy Wireless client provides access to AR System applications from wireless devices such as cell phones. The wireless signals from a client go to a gateway supplied by a third party, which translates the wireless input into HTTP requests.

BMC Remedy Administrator

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

BMC Remedy Import

Used to load external data into the AR System database. For instance, 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 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. An example of the interface is shown in Figure 2-4. BMC Remedy Import is available for Windows.

BMC Remedy Alert

Alerts users to events. For example, it can display a message alerting help desk personnel that a new problem has been assigned to them. Clicking on 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. Figure 2-5 shows an example of the interface. From BMC Remedy Alert, you can open up a list of alerts within 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."

Figure 2-3: BMC Remedy Administrator Interface

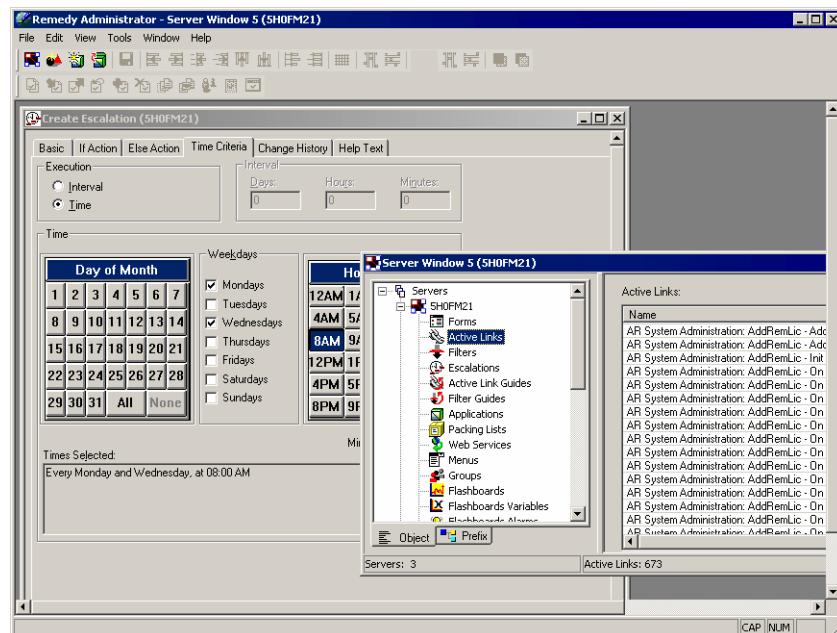
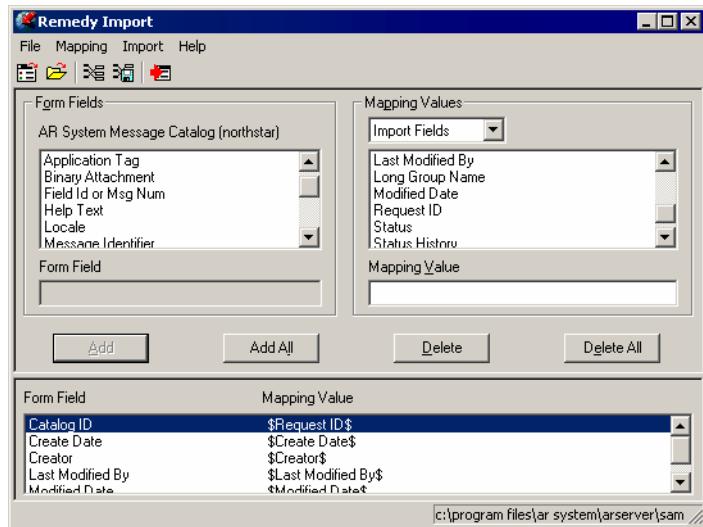
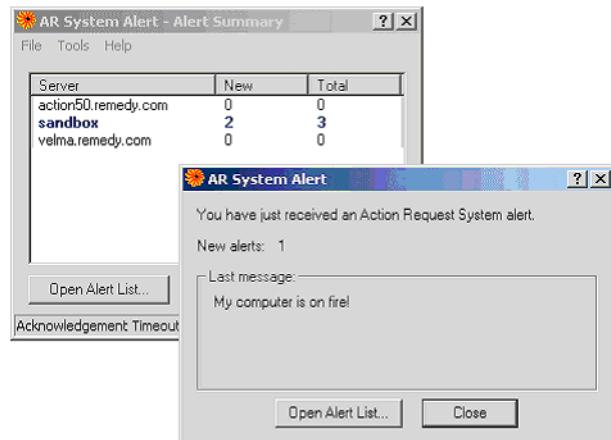


Figure 2-4: BMC Remedy Import Interface**Figure 2-5: BMC Remedy Alert Interface**

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.

Note: The BMC Remedy Mid Tier requires a supported Java Server Pages (JSP) engine. ServletExec 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://supportweb.remedy.com>.

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. 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). This API is one possible way to integrate with AR System.

The AR System server processes the data that is entered 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 transaction that is requested, enforcing any access control that has been 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.

The key components of the AR System server are:

- **arserverd process**—The primary AR System server process. It handles requests from the clients and interacts with the database.
- **arplugin process**—A companion process to the AR System server. It loads configured plug-in modules to interface with external data while processing vendor forms, validate users with external authentication sources, and extend 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 using remote procedure calls (RPCs) on top of a TCP/IP transport stack. The type of RPC is the Sun Microsystems 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 as well as 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 multi-threaded. This means that the database can perform multiple transactions simultaneously. In AR System, the `arserverd` server process is also multi-threaded. 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

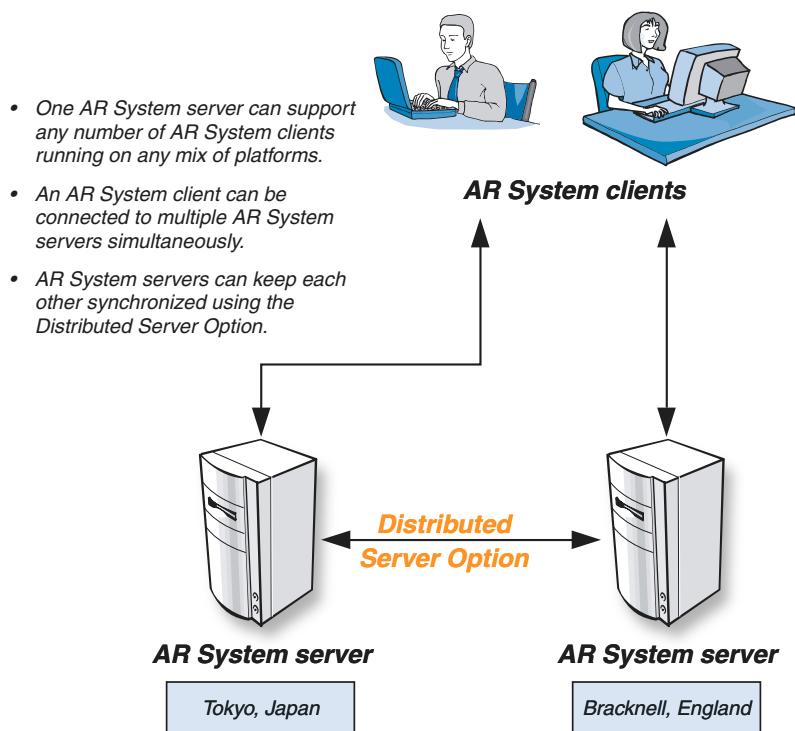
In an AR System environment, one AR System server can theoretically support any number of AR System client connections (limited by network bandwidth and server host and database performance). The clients can be on any mix of platforms.

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 allows 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 2-6: Many-to-many architecture



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 Administrator, 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 new 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 will store, 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 page 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 pressing 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 instance, the qualification to enforce the priority requirements in the previous example would be something like:

```
$Priority$ = "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 multi-tier 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. Once 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.

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 the BMC Remedy Administrator tool. 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 their workstation, and the groups he or she is a part of do not have View access to some of the fields, those fields will not be 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. Note that 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 within 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.

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

The following topics are provided:

- Where to integrate (page 38)
- Multi-platform issues (page 40)
- Choosing an implementation method (page 40)
- Integration technologies (page 41)

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 be accessing 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 about the AR System database, see the *Database Reference* guide.

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.

Multi-platform issues

A major consideration for every integration implementation is the range of platforms that will be 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 will be executed. In some cases, you can avoid the client functionality issue by running a process on the server from client workflow. Because the application will execute 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 will be 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 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 will almost always be used in conjunction with other tools and products to create an integrated solution. AR System has been designed to have many integration points, making it easy to combine with your other solutions.

The available technologies for integrating with AR System include:

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 45
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 55
Web Services	This integration technology (XML, WSDL, UDDI, and SOAP) allows you to build distributed applications without programming: <ul style="list-style-type: none"> • 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 75

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 115
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 dashboards) to AR System forms.	page 177
AREA Plug-Ins (AR System External Authentication)	BMC Remedy provides a special API that allows user logins to be validated using external processes, such as LDAP, Kerberos, or others. This API is called the AR System External Authentication (AREA) API.	page 115 and page 139
Filter Plug-Ins	The filter API allows you to use filters to permit other applications to make calls back into AR System.	page 115
ARDBC Plug-Ins (AR System Database Connectivity)	AR System Database Connectivity allows 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 runtime and implement calls that are analogous to the set, get, create, delete, and get-list API calls for entries in a form.	page 115 and page 139
Vendor forms	Vendor forms allow the 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 191
View Forms	View forms allow direct read-and-write access to data located in database tables that are not owned by AR System. This allows direct access to these tables, as if they were owned within AR System, without programming, database replication, or synchronization.	page 197
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 205
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 209

Method	Description	Page
Enterprise Integration Engine (EIE)	The Enterprise 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 225
Command Line Interface (CLI)	A Command Line Interface (CLI) is available with most AR System clients. This allows a client to be started and passed a set of parameters so that it will either be in a specific state and display information or it will complete a process and exit with no user interface displayed.	page 227
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 249
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 253
OLE Automation	BMC Remedy User supports OLE Automation. It can be both an Automation server and Automation client. This allows AR System to send commands or data to other applications and to receive commands or data from other applications.	page 267
Dynamic Data Exchange (DDE)	BMC Remedy User supports DDE. It can be both a DDE client and server. This allows AR System to pass data to other Windows applications and to be started “in context” by other applications.	page 287
SNMP	You can use SNMP to manage complex networks through SNMP-compliant management consoles and monitor network devices.	page 319
Source Control	When you integrate source control with AR System, application developers can quickly see in BMC Remedy Administrator if an object is checked out of the source control database and who its current owner is.	page 337
Licensing Applications	Authorized integration system vendors (ISVs) can make their applications licensable at the application and user levels.	page 363

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 46)
- Understanding the AR System API (page 47)
- Program structure (page 49)
- Multithreaded API clients (page 49)
- Using the API for integration (page 51)
- Issues and considerations (page 53)

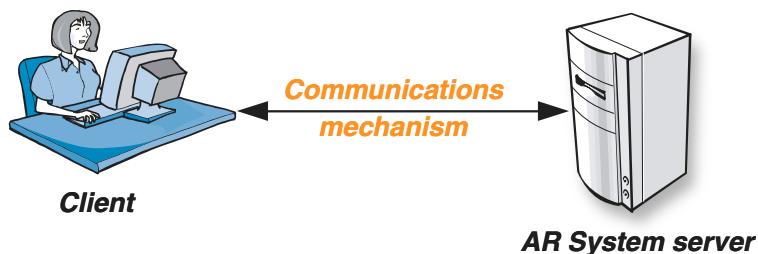
Overview

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



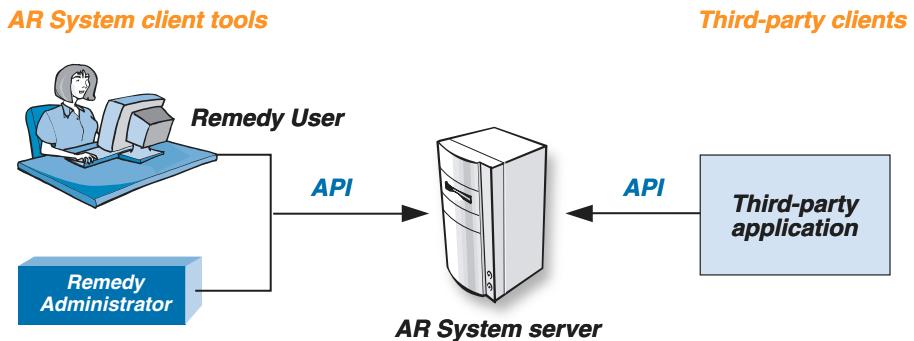
Typically, the client and server components are on different computer systems, and the communications mechanism is based on some form of networking technology, but it is possible to have both the client and server on the same host simply communicating between processes.

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 might be 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 *CAPI Reference* guide. 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 API is shipped on the product distribution media as an installation option. There is no additional license fee or charge for using the API to develop custom client applications, nor is there a license fee or charge for redistributing or selling such custom clients.

BMC Remedy User, BMC Remedy Administrator, and BMC Remedy Import were all built using this API. Therefore, any functionality available with these clients can be replicated in another client program.

The functions provided in the API library 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 a new 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* guide.
- **Shutdown/Cleanup**—Call `ARTermination` to perform environment-specific cleanup routines and disconnect from the AR System servers (required).

If you are using 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 present in the control record for all subsequent API function calls intended to operate within that session. Operations for a session are not restricted to a single thread; however, each session can only be active on one thread at any one 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 */
    AREEntryIdList entryId;
    entryId.numItems   = 1;
    entryId.entryIdList = (AREEntryIdType *) calloc(1,
sizeof(AREEntryIdType));
    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);

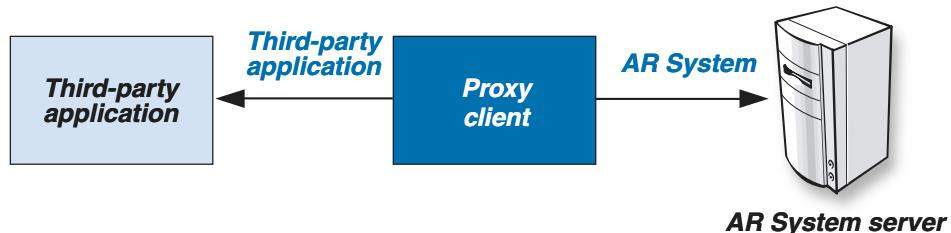
return 0;
}

```

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.

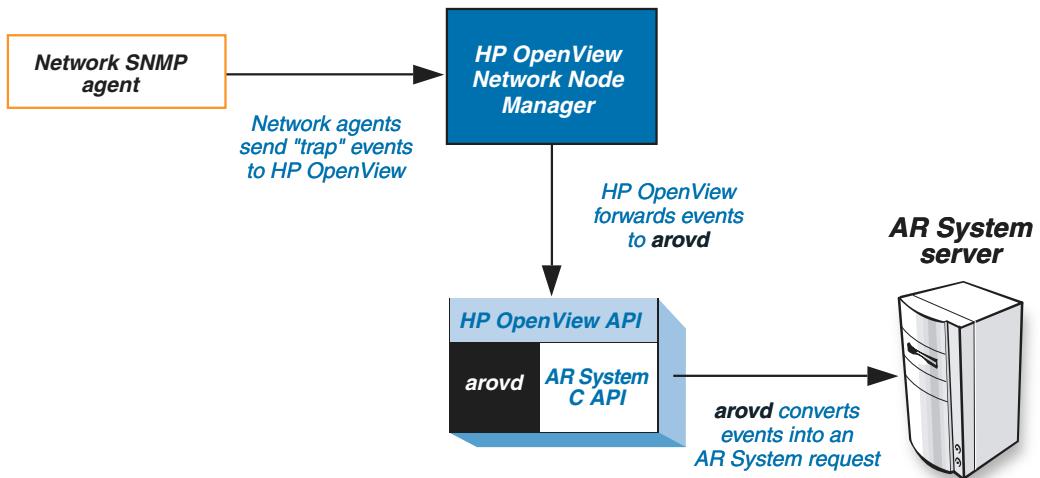
Example: Network management platform integration accessories

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 Sun 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



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 most likely need to make changes to your programs. The main program structure and processing, however, need not change.

For details about the new and changed calls in the API libraries, see the *AR System Release Notes* for the AR System versions newer than your integration. Also see the *C API Reference* guide for specific information about which API calls have been updated or replaced.

Chapter

5 Java API

The AR System Java API layer is a collection of classes, interfaces, and relationships that you can use to communicate with the AR System server. The Java API layer is designed to provide functionality equivalent to that provided by the AR System C API. This section provides an overview of the AR System Java API.

The following topics are provided:

- Overview (page 56)
- Programming model (page 56)
- Object factories (page 57)
- Criteria objects (page 58)
- Criteria list objects (page 58)
- Cloning objects (page 58)
- Exception handling (page 58)
- Java API program structure (page 59)
- Naming conventions: how C structures and functions correspond to Java API objects (page 68)
- Installation and deployment (page 69)
- Issues and considerations (page 73)

Overview

The Action Request System Java API layer is an application programming interface that integrates with AR System. It was introduced in AR System 5.0 and, like the C API, is forward and backward compatible with other versions of AR System. The Java API is a collection of classes, interfaces, and relationships that abstract the AR System C API and hide some of its details, such as the memory management for data structures, managing connections, and so on. Using the Java development environment, you can use the AR System Java API to communicate with the AR System server without programming in C.

In addition, many of the Java API objects are classes modeled after the Enterprise Java Beans (EJB) requirements for objects. Most of the methods required for an object to be classified as EJB (for example, load or store) are implemented in the AR System Java API.

AR System Java calls are passed through a JNI (Java native interface) layer and then through the C API to the AR System server. Because of this layering, the AR System Java API is not “pure Java” and is not portable across platforms. The Java API is included in the installation of AR System for all currently supported platforms.

Programming model

The AR System Java API layer is designed to adhere to object-oriented methodology and to be functionally equivalent to the AR System C API. In this implementation, the AR System server objects are treated as objects in the AR System Java API layer. All functionality of the AR System C API is represented as discrete objects used to create, retrieve, and manipulate AR System server objects and entries submitted by users (or other client programs) in the database.

Note: There is not a one-to-one correspondence in the mapping between the Java API and the C API.

You can also use the AR System Java API to manipulate data in the database. With the Java API you can manipulate objects such as schemas, workflow, menus, and so on. You can also use the Entry classes to manipulate entries in a form in different ways. For example, you can use the Java API to get a list of entries.

Object factories

In the Java API, server objects are exposed as Java objects. As a programming convenience, the AR System Java API uses factory design patterns to create, destroy, and retrieve server objects.

Each type of AR System object has its own object factory class, for example, ActiveLinkFactory and SchemaFactory. Each server object factory contains the newInstance() method for creating instances of objects. You must use the releaseInstance(Object obj) method to release all instances of any objects that you have created.

Each *<Object>Factory* contains three different find methods (find, findByKey, findObjects) to query for server object instances that contain data retrieved from AR System. In this way, you use the factory class methods of a server object to create new objects, or you use the find methods of the factory classes to search for existing objects on the AR System server. You use the get and set functions to allow you either to retrieve (getSchemaType) or to set (setSchemaType) data in that object. Then, you use the following methods to perform various operations on the AR System server:

- `create()`—Creates object on server.
- `store()`—Saves object on server.
- `remove()`—Removes object on server.
- `load()`—Fetches information about object from server.

Criteria objects

Criteria objects are used to evaluate and screen the results coming back to your Java program from AR System. Criteria objects let you define what attributes you want to retrieve. All server objects (workflow objects like active links, filters, and escalations; entry objects; support files; menus; fields; forms; and view objects) have criteria objects that allow you to specify which attributes to retrieve. For example, using the `ActiveLinkCriteria` object, you can specify name, execution order, timestamp information, object properties, and so on. Criteria objects follow this naming convention: `<Object>Criteria` (for example, `ContainerCriteria`, `FilterCriteria`, `EntryCriteria`, and so on).

Criteria list objects

All server object lists that can be retrieved in the AR System C API have a corresponding criteria list object as well. The list criteria object abstracts the query information that is applicable to a collection of objects.

Criteria objects for retrieving lists follow this naming convention: `<Object>ListCriteria` (for example, `ContainerListCriteria`, `EntryListCriteria`, and so on).

The exception to this naming convention is with workflow objects (active links, filters, and escalations), which use a common list criteria class called `WorkFlowObjectListCriteria`.

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 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.

Java API program structure

All AR System Java API programs follow the same basic outline. To implement your programs, use the following outline for your Java program:

- 1 Import the packages. (Import the `com.remedy.arsys.api` package as well as other packages you might be using in your program.)
- 2 Create a new `ARServerUser`. This corresponds to step 1 for the C API, except that you can create several `ARServerUser` instances, each of which is set up to talk to different servers or as different users.
- 3 Use the server object factory class to create an instance of the server object you will work with. Employ the `set` methods to describe and label the server object.
- 4 Set the context for the operation. This corresponds to choosing the `ARServerUser` object (user, password, and server information) for the operation or operations that follow.

Note: This step illustrates another difference between the C API and the Java API. In the C API, the control argument in each call to the server provides access information (user and server information as well as initialization information) on a per-call basis. In the Java API, you provide the user and server information separate from the calls you will make for that operation.

- 5 Perform the operation. Now, you can create entries or manipulate existing entries in the AR System server you specified in the context (the previous step).

6 Release the instance using its factory.

Because Java handles the deallocation of memory automatically (using garbage collection), another key difference between the C API and the AR System Java API is that when references to an object no longer exist, that object is assumed to be no longer needed. Then the memory occupied by that object is reclaimed. So, steps 3 and 4 of the general structure of a C API program do not have a counterpart in an AR System Java API program. However, in an AR System Java API program, you *must* use the clear method to release context (user and server) information as well as other methods of the classes you use, and you must use the releaseInstance method of the factory object to reclaim memory that has been allocated to a factory object. The underlying resources are not touched by garbage collection. They *must* be released.

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

```
package javaAPITest;
import com.remedy.arsys.api.*;
import java.io.*;

public class JavaAPITest {
    ARServerUser userInfo;
    Entry entry;
    NameID formName=new NameID("JavaAPITest");

    public static void main(String[] args) {
        JavaAPITest test=new JavaAPITest();
        test.connect();
        test.init();

        test.createRecordReplace
            ("Demo","1","nnnnn\nrrrrrr\rbothboth\r\nend");
        test.createRecord("Demo","1","test1");
        test.createRecord("Demo","2","test2");
        String entryID=test.createRecord("Demo","3","test3");
        test.modifyRecord(entryID);
        test.queryRecordsByID(entryID);
        test.queryRecordsByQual
            ("(\ 'Create Date\') > \"1/1/2004 12:00:00 AM\" )");
        test.queryRecordsByQual("(\ 'Create Date\') > \"1/1/2010\" )");
        test.queryRecordsByQual("This is a bad qualifier");
        test.cleanup();
    }

    public void connect() {
        System.out.println("Connecting to AR Server...");
        userInfo = new ARServerUser( );
        userInfo.setServer("Sandbox");
```

```

userInfo.setUser(new AccessNameID("Demo"));
userInfo.setPassword(new AccessNameID(""));
try {
    userInfo.verifyUser(new VerifyUserCriteria());
} catch (ARException e) {
    //This exception is triggered by a bad server, password or,
    //if guest access is turned off, by an unknown username.
    System.out.println("Error verifying user: "+e);
    //Clear memory used by our user context object
    userInfo.clear();
    System.exit(1);
}
System.out.println("Connected to AR Server.");
}

public void init() {
    EntryFactory entryFactory = EntryFactory.getFactory();
    entry = (Entry)entryFactory.newInstance();
    entry.setContext(userInfo);
    entry.setSchemaID(formName);
    System.out.println("Initialized.");
}

public String createRecordReplace
    (String submitter, String status, String shortDesc) {
String entryIdOut= "";
try {
    EntryFactory entryFactory = EntryFactory.getFactory();
    Entry entryr = (Entry)entryFactory.newInstance();
    entryr.setContext(userInfo);
    entryr.setSchemaID(new NameID("DMF:Test Replace"));

    FieldID[] fields= new FieldID[2];
    fields[0] = new FieldID(2); //Submitter
    fields[1] = new FieldID(7); //Status

    Value[] values= new Value[2];
    values[0] = new Value(submitter);
    values[1] = new Value(status);

    EntryItem[] entryItems= new EntryItem[3];
    entryItems[0] = new EntryItem(fields[0],values[0]);
    entryItems[1] = new EntryItem(fields[1],values[1]);
    entryItems[2] = new EntryItem(new FieldID(8),
        new Value(shortDesc)); //Short Description
    //#2 is done in one line just to show another method.

    entryr.setEntryItems(entryItems);
    entryr.create();
    entryIdOut =entryr.getEntryID().toString();
    System.out.println("Entry created, id #"+entryIdOut);
}

```

```
        } catch (ARException e) {
            ARExceptionHandler(e,"Problem while creating entry: ");
        }
        return entryIdOut;
    }

    public String createRecord
        (String submitter, String status, String shortDesc) {
        String entryIdOut= "";
        try {
            FieldID[] fields= new FieldID[2];
            fields[0] = new FieldID(2); //Submitter
            fields[1] = new FieldID(7); //Status

            Value[] values= new Value[2];
            values[0] = new Value(submitter);
            values[1] = new Value(status);

            EntryItem[] entryItems= new EntryItem[3];
            entryItems[0] = new EntryItem(fields[0],values[0]);
            entryItems[1] = new EntryItem(fields[1],values[1]);
            entryItems[2] = new EntryItem(new FieldID(8),
                new Value(shortDesc)); //Short Description
            //#2 is done in one line just to show another method.

            entry.setEntryItems(entryItems);
            entry.create();
            entryIdOut =entry.getEntryID().toString();
            System.out.println("Entry created, id #"+entryIdOut);
        } catch (ARException e) {
            ARExceptionHandler(e,"Problem while creating entry: ");
        }
        return entryIdOut;
    }

    void modifyRecord(String eidStr)
    {
        try
        {
            EntryID entryID = new EntryID(eidStr);
            EntryKey entryKey = new EntryKey(formName, entryID);

            entry=EntryFactory.findByKey(userInfo, entryKey, null);

            EntryItem[] entryItems= new EntryItem[1];
            entryItems[0] = new EntryItem(new FieldID(8),
                new Value("Modified by JavaAPItest"));

            entry.setEntryItems(entryItems);
            entry.store();
        }
    }
}
```

```

        System.out.println("Record #"+eidStr+" modified
successfully.");
    }
    catch(ARException e)
    {
        AREExceptionHandler(e,"Problem while modifying record: ");
    }
}

void queryRecordsByID(String eidStr)
{
    System.out.println("Retrieving record with entry ID#"+eidStr);
    try {
        EntryID entryID = new EntryID(eidStr);
        EntryKey entryKey = new EntryKey(formName, entryID);

        entry=EntryFactory.findByKey(userInfo, entryKey, null);

        EntryItem[] entryItemList = entry.getEntryItems();
        if( entryItemList == null )
        {
            System.out.println("No data found for ID#"+eidStr );
            return;
        }

        System.out.println("Number of fields: " +
                           entryItemList.length );

        // Set the field criteria to retrieve all field info
        FieldCriteria fCrit = new FieldCriteria( );
        fCrit.setRetrieveAll(true);

        for( int i = 0; i < entryItemList.length; i++ )
        {
            // Set the Field ID to retrieve
            FieldID id = entryItemList[i].getFieldID();
            FieldKey key = new FieldKey( formName, id );

            // Get the field's details
            Field field = FieldFactory.findByKey( userInfo, key,
                                                fCrit );

            // Output field's name, value, id, and type
            System.out.print(field.getName().toString());
            System.out.print(": " + entryItemList[i].getValue());
            System.out.print(" ID: " + id);
            System.out.print(" Field type: " +
                           field.getDataType( ).toInt());
        }
    }
}

```

```
// Handle if Date..
if ( field.getDataType().toInt() == 7 )
{
    System.out.print(" Date value: ");
    Timestamp callDateTimeTS =
        (Timestamp)entryItemList[i].getValue().getValue();
    //First getValue returns a Value object,
    //second returns a generic Object that
    //can be cast as a Timestamp
    if (callDateTimeTS != null)
        System.out.print(callDateTimeTS.toDate());
    }
    System.out.println("");
    field.clear();
    //Release memory held by our field object
}   //for each entryItem
}
catch( AREException e )
{
    AREExceptionHandler
        (e,"Problem while querying by entry id: ");
}
}

void queryRecordsByQual(String qualStr)
{
    System.out.println
        ("Retrieving records with qualification "+qualStr);
    try {
        // Set the field criteria to retrieve all field info
        FieldCriteria fCrit = new FieldCriteria( );
        fCrit.setRetrieveAll(true);

        // Retrieve all types of fields
        FieldListCriteria fListCrit =
            new FieldListCriteria(formName, new Timestamp(0),
            FieldType.AR_ALL_FIELD);

        // Load the field array with all fields in the test form
        Field[] formFields = FieldFactory.findObjects(userInfo,
            fListCrit, fCrit);

        // Create the search qualifier.
        QualifierInfo myQual = Util.ARGetQualifier(
            userInfo, qualStr, formFields, null,
            Constants.AR_QUALCONTEXT_DEFAULT);

        FieldFactory.getFactory().releaseInstance( formFields );

        // Set the EntryListCriteria
        EntryListCriteria listCriteria = new EntryListCriteria( );
        listCriteria.setSchemaID( formName );
        listCriteria.setQualifier( myQual );
    }
}
```

```

// Define which fields to retrieve in the results list
// Note that the total width including separators must be
// <256 characters
// Here the total width=15+15+25+3*1=58
// 3*1 is three separators of 1 character each
EntryListFieldInfo[] entryListFieldList =
    new EntryListFieldInfo[ 3 ];
// Submitter(2)
entryListFieldList[0] = new EntryListFieldInfo
    (new FieldID(2), 15, " " );
// Status(7)
entryListFieldList[1] = new EntryListFieldInfo
    (new FieldID(7), 15, " " );
// Short Description(8)
entryListFieldList[2] = new EntryListFieldInfo
    (new FieldID(8), 25, " " );

// Set the entry criteria
EntryCriteria criteria = new EntryCriteria( );
criteria.setEntryListFieldInfo( entryListFieldList );

// Make the call to retrieve the query results list
Integer nMatches = new Integer(0);
// AR Java API 5.1 syntax
//     EntryListInfo[] entryInfo = EntryFactory.findEntryListInfos(
//         userInfo, listCriteria, criteria, nMatches );
// AR Java API 6.0 syntax adds the boolean useLocale flag.
// Set it to false to return all entries regardless of locale.
EntryListInfo[] entryInfo = EntryFactory.findEntryListInfos(
    userInfo, listCriteria, criteria, false, nMatches );

System.out.println
    ("Query returned "+ nMatches +" matches.");
if( nMatches.intValue() > 0 )
{
    // Print out the matches
    System.out.println("Request Id      Submitter"+
        "      Status      Short Description" );
    for( int i = 0; i < entryInfo.length; i++ )
        System.out.println
            (entryInfo[i].getEntryID(
).toString( ) +
                "      " + new String(
                    entryInfo[i].getDescription( )));
}
EntryFactory.getFactory().releaseInstance( entryInfo );
}
catch( AREexception e )
{
    AREExceptionHandler
        (e,"Problem while querying by qualifier: ");
}
}

```

```
public String getString(String prompt)
{
    try {
        BufferedReader in = new BufferedReader(
            new InputStreamReader(System.in));
        System.out.print(prompt);
        return in.readLine();
    }catch(IOException e)
    {
        System.out.println("Error getting input: "+e);
        return (String)null;
    }
}

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

public void printStatusList(StatusInfo[] statusList) {
    if (statusList == null || statusList.length==0) {
        System.out.println("Status List is empty.");
        return;
    }
    System.out.print("Message type: ");
    switch(statusList[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[0].getMessageType()+"')");
            break;
    }
    System.out.println("Status List:");
    for (int i=0; i<statusList.length; i++) {
        System.out.println(statusList[i].getMessageText());
        System.out.println(statusList[i].getAppendedText());
    }
}
```

```
        }
    }

public void cleanup()
{
    //Clear memory held by our entry object
    //Similar cleanup should be done for all objects created by a
    //factory
    EntryFactory.getFactory().releaseInstance(entry);
    //Clear memory used by our user context object
    userInfo.clear();
    System.out.println("AR System objects cleaned up.");
}
```

Naming conventions: how C structures and functions correspond to Java API objects

Developers who are familiar with the AR System C API should be able to find their way in the Java API. Some common terms in C have their parallels in Java, as outlined in the following table:

Table 5-1: Differences in AR System API naming conventions

C API	Java API	Differences
AR<object>Struct (see exceptions that follow)	<object>Info	In the Java API the AR at the beginning of C functions is left out. For example, the names of functions associated with ARGetActiveLink are renamed to the ActiveLink class with its associated member functions used to get, set, or create active links. Because there are no data structures as such in the Java environment, data structures in the C API translate to classes in the AR System Java API with the suffix - Info. For example, the ARQualifierStruct data structure translates to the QualifierInfo class.
AR<object>List	<object>Info (an array)	Because lists are handled as arrays in the Java API, the - List suffix becomes the - Info suffix. For example, ARPermissionList from the C API corresponds to the PermissionInfo class in the AR System Java API. The exceptions to these naming conventions pertain to the classes used for retrieving lists of entries and lists of fields. For example, in the C API, you would use the AREntryListList to return a list of entries as concatenated strings. In the Java API, you would use the find and findObjects factory methods to retrieve a list of entries as concatenated strings. For more information, see “Java API program structure” on page 59.
NameType	NameID	The C API suffix -Type becomes - ID in the Java API, so that a variable NameType becomes NameID.
Get	load, findByKey	Where the C API uses functions, the Java API uses methods on the associated object. So instead of ARGetField, the Java API will be called by field.findByKey().
Set	store	None
Create	create	None

Table 5-1: Differences in AR System API naming conventions

C API	Java API	Differences
Delete	remove	None
GetList	find, findObjects	None

Exceptions to naming conventions

The ARControlStruct is an exception to the naming conventions because user information required to perform AR System operations is handled slightly differently in the Java API. The ARServerUser class is used to handle user login information for object operations. You specify user information through the ARServerUser class when you perform a setContext operation for a specific class of server object.

Installation and deployment

To use the AR System Java API, you must have the AR System C API library files and dlls installed on the same machine where you run the AR System Java API. See the *C API Reference* guide for more information.

Contents of the AR System java object installation

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

Component	Description
arapi70.jar arutil70jni.jar	JAR files that contains the API class files
arjni70.dll (Windows) arutil70.dll (Windows)	Java Native Interface (JNI) implementation
libarjni70.s1 (HP-UX) libarutiljni70.s1 (HP-UX)	
libarjni70.a (AIX) libarutiljni70.a (AIX)	
libarjni70.so (Solaris) libarutiljni70.so (Solaris)	
ardoc70.jar	Javadoc-generated HTML docs for the API
JavaDriver.java InputWriter.java OutputWriter.java	Sample Java code file that shows examples of using the Java API
arapi70.dll, arutil70.dll, arrpc70.dll (Windows only)	The current AR System API development kit (the debug versions of each dll are also included)

Windows environment setup

Implementations of the JDK vary depending on environment, but you might want to check the following components:

Component	Description
Compiler	Requires JDK 1.2.2 (or higher). To install the JDK, use the instructions provided by Sun Microsystems.
JAR Files and CLASSPATH	Arapi70.jar should be present in CLASSPATH. If you are running the AR System Java API on a web server, you must set the ClassPath information about the web server.
.dlls and System Path	The Java API requires the AR System API dlls. <i>All</i> dlls should be available in the dll search path. Set arrpcXX.dll, arutlXX.dll, and arapiXX.dll in the System path. (You can either add <ar_install_dir> to your path or put your executable file in <ar_install_dir>.)

Note: XX in the file names stands for the AR System version; for example, 70 would be inserted for AR System 7.0.

UNIX environment setup

Implementations of the JDK vary depending on environment, but you might want to check the following components:

Component	Description
Compiler	Requires JDK 1.2.2 (or higher). To install the JDK, use the instructions provided by Sun Microsystems.
JAR Files and CLASSPATH	<code>arapiXX.jar</code> should be present in <code>CLASSPATH</code> . If you are running the AR System Java API on a web server, you must set the <code>CLASSPATH</code> information about the web server.
PATH	Update your <code>PATH</code> environment variable to include the bin directory for the JDK according to the installation instructions from Sun.
Library Path	For Solaris and IBM AIX, set the <code>LD_LIBRARY_PATH</code> environment variable to the location of where <code>libarjniXX.so</code> is installed. For HP-UX, set the <code>SHLIB_PATH</code> environment variable to the location of where <code>libarjniXX.s1</code> is installed.

Issues and considerations

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

- For more information about the Java API, Java file names, or file locations, see the Java API documentation HTML files in the `ardocXX.jar` file located in the `\ArServer\api\doc` folder. To access the Java API documentation, you must unzip the contents of the `ardocXX.jar` file. (To unzip a `.jar` file, use the Java `jar` executable, which is located in the `.bin` directory where the Java JRE is installed. Enter `jar -xvf \ARServer\api\doc\ardocXX.jar`.) Then, navigate to the `javadoc` folder, and click the `index.html` file to see an overview of the entire AR System Java API documentation.
- Test your environment by running the AR System server and the `JavaDriver.java` program. This program illustrates the capabilities of the Java API. The `JavaDriver.java` program works almost exactly the same as the C driver program. For more information, see the *C API Reference* guide.
- When running a program that uses the API, make sure that `arjniXX.dll` can be found in the `.dll` search path.
- If you are running your application from the command line, make sure to include the `classpath` option when you specify the Java command.

For example, the following command runs the `JavaDriver.java` program and the `classpath` option points to the current directory for the JAR file:

```
java -D java.library.path = <directory>; -classpath .;arapi70.jar  
JavaDriver
```


Chapter

6 Web services

This section provides information about using AR System and web services to connect to other applications across the internet or an intranet. It describes how to publish AR System functionality as a web service and how to invoke web services created by external applications to push data to these external applications or to make the data from these external applications available through the AR System.

The following topics are provided:

- Overview of web services (page 76)
- Setting up your environment for web services (page 81)
- Creating a web service (page 87)
- Publishing a web service (page 99)
- SOAP headers and authentication (page 103)
- Consuming web services (page 105)
- Web services limitations (page 110)

Overview of web services

A web service provides an interface that enables messages to be sent to and received from an application over a network (Internet or intranet), using standard Internet technologies. AR System web services use a combination of protocols such as HyperText Transfer Protocol (HTTP) and Extensible Markup Language (XML), which are platform independent, thus enabling the application to be accessed regardless of its hardware, software platform, or programming languages.

You *publish* web services to make AR System functionality, like submit, modify, and query, available over the web. You *consume* web services when you use AR System applications to access third-party web services.

You use BMC Remedy Administrator to create and publish AR System web services to be used and consumed by external applications.

Web service architecture

For both publishing and consuming web services, you specify a base form to which the information is set, or through which the information is pushed to other forms or applications. You map the AR System fields on your base form to input or output parameters of a web services operation. A field can participate as either an input parameter, an output parameter, or both. You can map to a simple flat document or a complex hierarchical document involving parent and child relationships.

AR System web services make AR System operations like submit, modify, and query available to other applications regardless of hardware or software platforms and independent of the programming languages in which the applications are written. You use BMC Remedy Administrator to create AR System Web Service objects containing AR System data. You then map AR System fields to web service parameters and specify what types of operations can be performed by the web service. Typically, these operations include Create, Set, Merge, Delete, or GetList. After you create the web service, its mapping definition is stored on the AR System server.

The web service is automatically deployed to the mid tier when the URL is typed into the address field in a browser. A Web Services Description Language (WSDL) document for the web service is created in the mid tier. This WSDL file is an XML-based document that describes the web service. Apache AXIS, a third-party web service server will be installed with the mid tier. The mid tier also includes new Java classes and new API calls.

The base form

Each web service has a base form on which it operates. You specify form this when you create your web service. When you create a web service by right-clicking on a form name in the server window, the Base Form field is automatically filled for you. 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 across multiple AR System forms. For multiple AR System forms, the base form is the master form.

Operations

Each web service has a list of operations. You can rename these operations, delete some operations, or even create new operations, but they must be one of these three types: Create, Get, or Set. You can have more than one operation of the same type, or you can have no operations of a particular type.

When you create a web service, by default it automatically has four named operations:

- OpCreate (of the type Create)
- OpGet (of the type Get)
- OpGetList (of the type Get)
- OpSet (of the type Set)

For more information, see Appendix A, “Web service operation types.”

Mapping

Each operation has an “input mapping” and an “output mapping.” A mapping describes how individual elements of the incoming and outgoing XML document are mapped to fields and forms of the AR System. These are essentially the input and output parameters of the web service. Mappings are edited through the mapping dialog box of BMC Remedy Administrator; for mapping procedures, see “Mapping to simple and complex documents” on page 384. When you create a web service, default input and output mappings are provided for each of the four default operations: OpCreate, OpGet, OpGetList, and OpSet. You can change the mapped fields. You can also change the name of the XML element that a field is mapped to, and you can create more XML elements.

XML schema

Each web service can be associated with an XML schema (.xsd file). Global elements and complex types referred to in the schema can be used in mappings associated with operations. For more information, see “XML editing” on page 397.

Basic web services

A basic AR System web service has four operations: OpCreate, OpSet, OpGetList, and OpGet. For the fields on your base form, the system provides an XML compliant element name and maps it to an input and output parameter, where applicable. These XML compliant names are used in the generation of the Web Service Description Language (WSDL) file for the web service. An external client can call this web service and create, modify, or get records from your form.

Custom web services

You can change some of the items to customize your web service, as follows:

- Rename the Web Service—The default name is the same as the base form name.
- Rename the operations—The default names are OpGet, OpGetList, OpSet, and OpCreate.
- Remove operations and add new ones—The four operations named above are added by default.
- Remove fields and add new ones to the mapping—Some fields are present by default.
- Include only some parts of special fields in your mapping—All parts are present by default. An example would be attachment fields with multiple parts such as attachmentName, attachmentData, and attachmentOrigSize. You can select only those parts that you require.
- Rename XML element names—BMC Remedy Administrator 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—Examples would be to group together certain fields to make “complex-types” or “SOAP-structures,” or designate a field as an attribute rather than a sub-element.
- Select the message system for the web service communication—RPC/encoded, RPC/encoded with first parameter as XML string, or document/literal.
- Specify an external XML schema and use global elements/complex types in various operations.

Messaging protocols

AR System web services use a combination of protocols such as HyperText Transfer Protocol (HTTP) and Extensible Markup Language (XML), which are platform independent, thus enabling the application to be accessed regardless of its hardware, software platform, or programming languages.

Simple Object Access Protocol (SOAP) is an HTTP/XML-based protocol and is used as the primary transport protocol for the messages shared by applications in web services. SOAP is an XML-based packaging format for the information being transferred, and also contains a set of rules for translating applications and platform-specific data types into XML.

Note: AR System web services implementation is based on SOAP 1.1 and WSDL 1.1 specifications from W3C.

Web services use two styles of messaging:

- **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.

Both these styles can be divided into literal or encoded substYLES. Literal means that the messages will be encoded according to the XML schema, and encoded means that the messages are constructed according to SOAP encoding rules. The resulting four possible styles and sub-style combinations are:

- RPC-Literal
- RPC-Encoded
- Document-Literal
- Document-Encoded

Note: For publishing, AR System supports *only* document/literal web services.

Further variations are possible, but they are not used in the AR System. The style you choose depends on your compatibility requirements. To communicate with MicrosoftDotNet, use Document-Literal. For everything else, you would typically use RPC-Encoded. For complex documents, use a literal format since AR System does not yet support arrays and structures in the encoded formats.

Web service description language (WSDL) file

Web Services Description Language (WSDL) is an XML-based language used to define web services and how to access them. For an example of a WSDL file, see Figure 6-7 on page 91.

Further information about SOAP and specifications for WSDL can be found at the following sources:

<http://www.w3.org/TR/SOAP/>
<http://www.w3.org/TR/wsdl>

Writing web service clients

You need to enable your client to communicate with your web service. Popular environments for writing web service clients are Microsoft.NET and Apache AXIS. These environments have tools that automatically generate code to invoke a web service, given the WSDL.

- In Microsoft.NET Visual Studio, autogenerate the invocation code by “adding a web reference.” When prompted for a URL, enter your WSDL URL.
- In AXIS, run `WSDL2java` from the command line with the WSDL URL as a command line parameter. The autogenerated code will be a class that has methods that correspond exactly to the operations you created in the web service.

Each method will have input and output parameters corresponding to the mappings you created. To invoke the web service, you need to instantiate the class and invoke a method with the correct parameters.

For more information, see the *Instructions for Creating Web Service Clients* white paper available from the Customer Support website at <http://supportweb.remedy.com>.

Setting up your environment for web services

The section details Java runtime environment considerations for setting up your environment for web services. It then discusses configuring AR System with Internet access through a proxy server.

Important: Select the web service option during the installation of the AR System server.

Java runtime environment considerations

AR System web services require that the Java Runtime Environment (JRE) on the system running BMC Remedy Administrator.

If you install the JRE *before* you install or upgrade the AR System server, the AR System server installer will automatically update your ar.cfg file and set your PATH.

If you install JRE *after* installing the AR System server, either reinstall the server, or add the following lines to your ar.cfg file (default location:

C:\Program Files\AR System\):

```
ARF-Java-Class-Path: C:\Program Files\AR System\axis-ant.jar
ARF-Java-Class-Path: C:\Program Files\AR System\axis.jar
ARF-Java-Class-Path: C:\Program Files\AR System\commons-discovery-0.2.jar
ARF-Java-Class-Path: C:\Program Files\AR System\commons-logging-1.0.4.jar
ARF-Java-Class-Path: C:\Program Files\AR System\jaxrpc.jar
ARF-Java-Class-Path: C:\Program Files\AR System\junit.jar
ARF-Java-Class-Path: C:\Program Files\AR System\log4j-1.2.8.jar
ARF-Java-Class-Path: C:\Program Files\AR System\saaj.jar
ARF-Java-Class-Path: C:\Program Files\AR System\websvc70.jar
ARF-Java-Class-Path: C:\Program Files\AR System\wsdl4j-1.5.1.jar
ARF-Java-Class-Path: C:\Program Files\AR System\wsdl4j.jar
ARF-Java-Class-Path: C:\Program Files\AR System\xercesImpl.jar
ARF-Java-Class-Path: C:\Program Files\AR System\xercesSamples.jar
ARF-Java-Class-Path: C:\Program Files\AR System\xmlParserAPIs.jar
#ARF-Java-VM-Options: -Dhttp.proxySet=true -
Dhttp.proxyHost=zermatt -Dhttp.proxyPort=3129
Plugin: WebService.dll
```

Also, if you install the JRE after installing the AR System server, add the JRE directory to your PATH, for example:

C:\Program Files\JavaSoft\JRE\1.4.x\bin\hotspot

Configuring AR System with Internet access through a proxy server

You can configure BMC Remedy Administrator and the AR System server with Internet access through a proxy server.

Note: If you insert any proxy configuration values into the ar.cfg file, you must stop and restart the server to pick up the new configuration information.

Accessing proxy configuration information

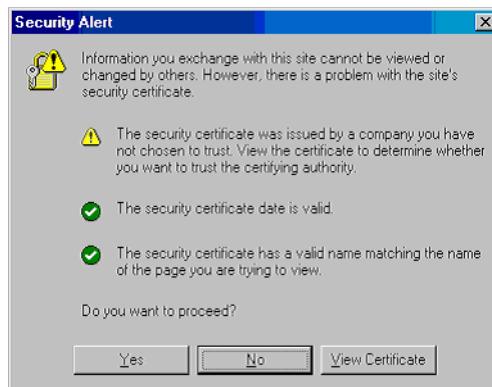
If your network administrator has already configured your browser (Mozilla or Internet Explorer) with proxy settings, you can view the values in the browser's configuration settings and insert the values in the ar.cfg file. AR System will not automatically retrieve the browser configurations. In Internet Explorer, choose Tools > Internet Options > Connections > LanSettings > ProxyServer.

Accessing WSDL or web service over https

Make sure that JRE 1.4.x is specified as the current JRE in the system where BMC Remedy Administrator and the web service filter plug-in are installed. The 1.4.x JRE installation includes JSSE.jar and related settings that enable Java programs to communicate over https without any manual changes in configuration files. If you must use previous versions of JRE, go to <http://java.sun.com> to learn more about copying JSSE-related jar files and making changes to configuration files.

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 local client.

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



Browsers will allow you to continue if you ignore warnings or errors; however, the 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 then continue with the AR System. In the second alert shown in Figure 6-1, you should update the certificate store to trust the specified company or certificate. See <http://java.sun.com/products/jssse/> or your local administrator to learn more about frequent issues with certificates.

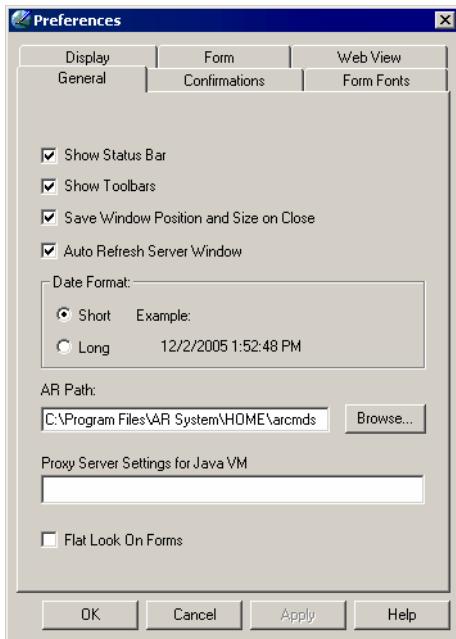
Configuring a plug-in

To configure a plug-in with Internet access through a proxy server, perform the following steps.

► To configure a plug-in

- 1 Open the server window.
- 2 Select a server to administer.
- 3 Choose File > Preferences.
- 4 Click the General tab.
- 5 Enter the following settings in the Proxy server settings for Java VM: field:

```
-Dhttp.proxySet=true -Dhttp.proxyHost=<host_name> -  
Dhttp.proxyPort=<port_number>
```

Figure 6-2: Configuring plug-in proxy settings for Java VM

- 6** Click **Apply** to save your changes.
- 7** Restart the server to pick up the new configuration information.

Configuring BMC Remedy Administrator

To configure BMC Remedy Administrator for Internet access through a proxy server (for example, creating filters that consume web services), perform the following steps.

► To configure BMC Remedy Administrator

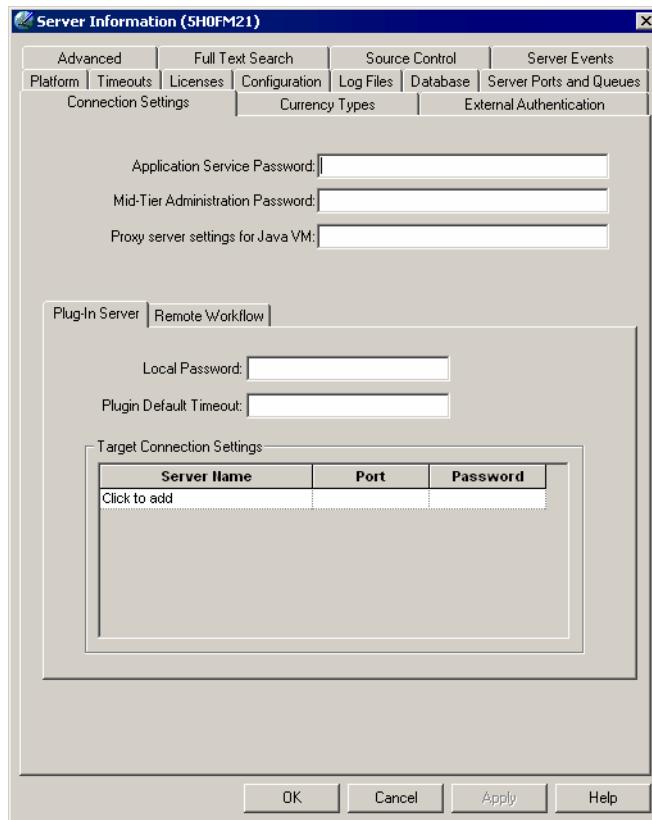
- 1** Open the server window.
- 2** Select a server to administer.
- 3** Choose **File > Server Information**.
- 4** Click the **Connection Settings** tab.
- 5** Enter the following settings in the **Proxy server settings for Java VM** field:

```
-Dhttp.proxySet=true -Dhttp.proxyHost=<host_name> -
Dhttp.proxyPort=<port_number>
```

If you have a socks proxy, use the following value instead:

-DsocksProxyHost=<host_name>-DsocksProxyPort=<port_number>

Figure 6-3: Configuring BMC Remedy Administrator proxy settings for Java VM



- 6 Click Apply to save your changes.

For more details about these options, see the following URL:

<http://java.sun.com/j2se/1.4.2/docs/guide/net/properties.html>

Other protocols

https

-Dhttps.proxySet=true -Dhttps.proxyHost=<host_name> -
Dhttps.proxyPort=<port_number>

socks

-Dssocks.proxyHost=<host_name> -Dssocks.proxyPort=<port_number>

Note: socks with Authentication is not supported.

If you have some internal hosts that should not use the proxy, set it as follows:

-Dhttp.nonProxyHosts="*.foo.com"

More information

For more details about these options, see <http://java.sun.com/j2se/1.4/docs/guide/net/properties.html>

Creating a web service

Creating a web service consists of creating a web service object, mapping AR System fields to web service parameters and specifying what types of operations can be performed by the web service.

The web service is created and modified in BMC Remedy Administrator using the Web Services Graphical User Interface.

You can create a basic or custom web service. See “Creating a basic web service” and “Creating a custom web service.”

Creating a basic web service

To create a web service using the default settings, right-click on a form to display the Create Web Service dialog box and save the web service.

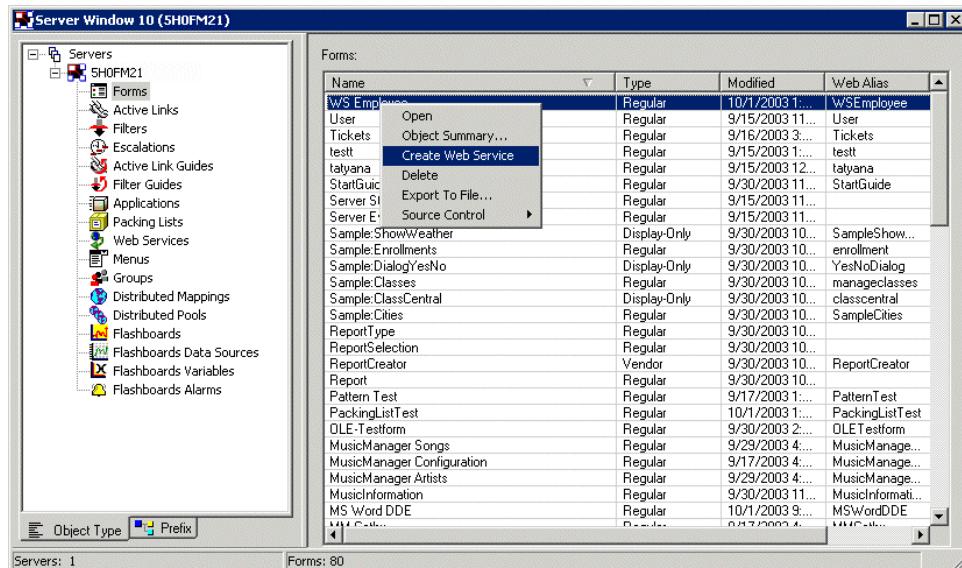
The resulting web service will have four operations: OpCreate, OpSet, OpGetList, and OpGet. For each of these default operations, BMC Remedy Administrator has already defined input and output parameters. For the fields on your base form, the system provides an XML compliant element name and maps it to an input and output parameter, where applicable. These XML compliant names are used in the generation of the WSDL file for the web service. An external client can call this web service and create, modify, or get records from your form.

► To create a basic web service using the defaults

- 1 Open a server window.
- 2 Select the appropriate server.
- 3 Create a form to use as your base form if you do not already have one.
- 4 Expand the list of server objects and select the Form object to display a list of available forms in the pane on the right.
- 5 Right-click on the form that you want to use as the base form (see Figure 6-4).

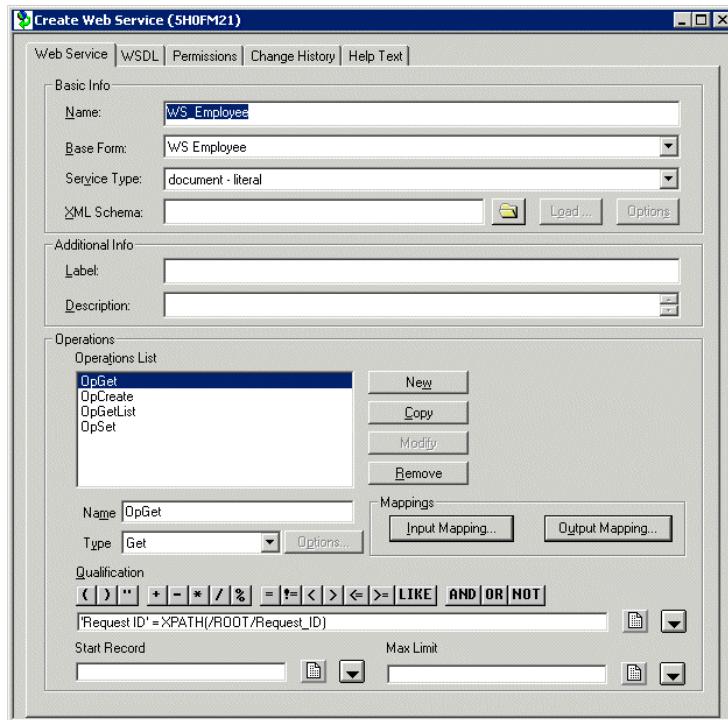
This is the form through which your web service operations will function.

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



- 6 Select Create Web Service from the pop-up list. The Create Web Service dialog box appears as shown in Figure 6-5.

Figure 6-5: Create web service dialog box



The default settings are as follows:

- The Name field is automatically filled in with the web service name based on your selected form name.
- The Base Form field is automatically filled in with the name of the selected form.
- The Service Type is the default document-literal.
- The XML Schema, Label and Description fields are blank.
- The default operations are displayed in the Operations List. They are OpCreate, OpGet, OpGetList, and OpSet.
- A qualification is automatically created for Get and Set operations. The qualification is listed at the bottom of the dialog box. You do not need a qualification for the Create operation.

- If you select the `OpGetList` operation, the Start Record and Max Limit fields are filled in. For more information about these fields, see “Setting the starting record and setting the maximum limit” on page 381.
 - The default XML complex types, elements and input and output mappings are automatically set for each operation.
- 7 Click the WSDL tab to see the URL for your Web Service Description Language (WSDL) file.

A sample URL for your WSDL file is displayed in the field.

Figure 6-6: Sample URL displayed in WSDL field



- 8 Complete the URL as appropriate for your configuration, as follows:
- a Replace `<midtier_server>` with the name of the web server where the mid tier is running.
 - b After WSDL, add /public or /protected depending on the web service’s permissions.

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

`http://POLYCARP/arsys/WSDL/public/POLYCARP/WS_Employee.`

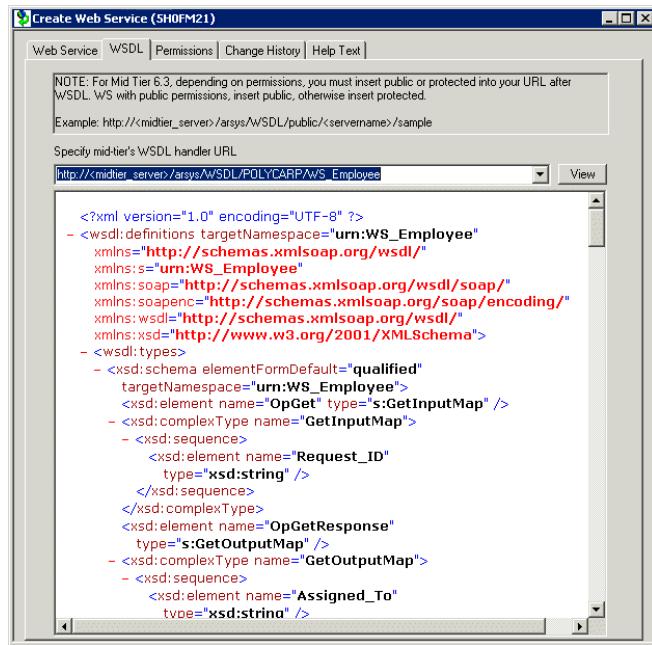
If the web service does not have public permissions, use:

`http://POLYCARP/arsys/WSDL/protected/POLYCARP/WS_Employee.`

If you are creating a new web service, you must save it before proceeding to the next step.

- 9 Click the View button to view your WSDL file in the window.

Figure 6-7: A WSDL file displayed in the WSDL tab of the web service dialog box



- 10 Set the Permissions, Change History, and Help Text as necessary. The permissions are the same whether the web service is visible or hidden.
Set the permissions to Public if you are going to publish your web service over an internet or intranet for general use.
 - 11 Save your web service.
- To enable your clients to communicate with your web service, see “Writing web service clients” on page 81.

Creating a custom web service

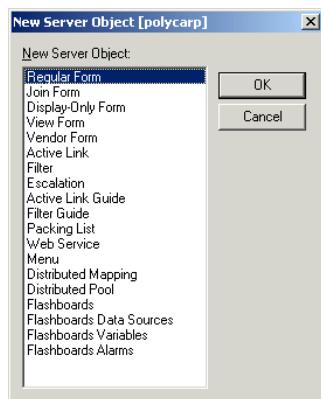
To create a custom web service, follow steps 1 through 6, and continue from step 7 on page 93. Or, use the New Server Object selection box, as described in the following procedure.

► To create a custom web service

- 1 Open a server window.
- 2 Select the appropriate server.
- 3 Choose File > New Server Object.

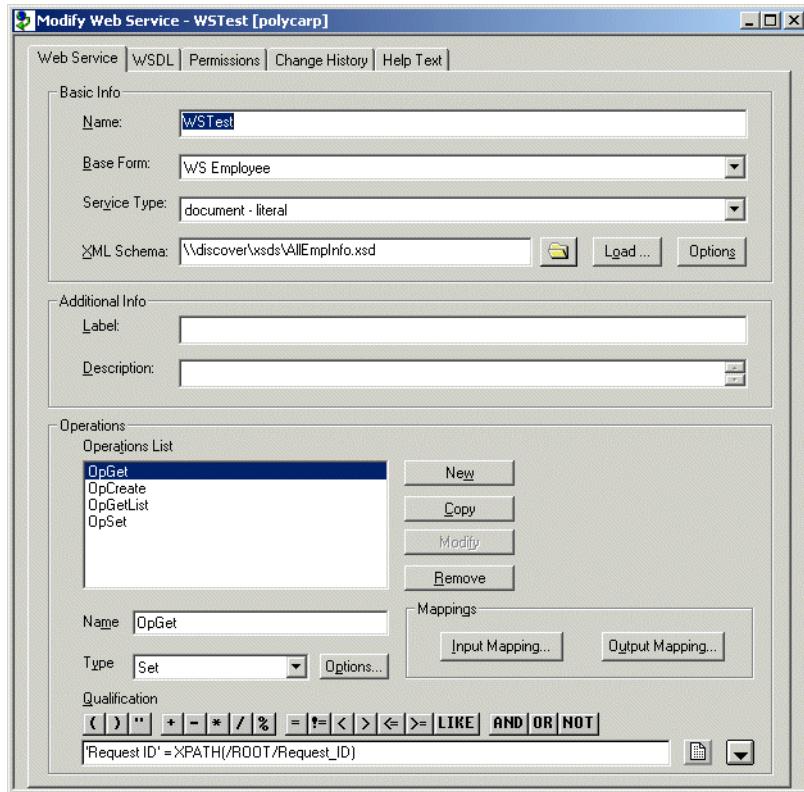
The New Server Object dialog box appears.

Figure 6-8: New server object dialog box



- 4 Select Web Service from the list.

The Create Web Service dialog box appears.

Figure 6-9: Modify web service dialog box

- 5** Enter the name of the web service in the Name field.

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

- 6** From the Base Form list, select the form that you are going to use.
This is the form through which your web service operations will function.
- 7** Select a Service Type from the list. See "Messaging protocols" on page 79 for more information.

- 8 In the XML Schema field, type in or browse to an external XML schema, or leave the field blank to use a default XML schema. The options are:

- To use an external XML schema, the elements or complex types contained in this file will be used for the mapping of AR System form fields to operation parameters. See “Importing an external XML schema” on page 405 for more information about using an external file.
If you selected an XML schema, go to step 9.
- If you leave the field blank, BMC Remedy Administrator will create an XML schema for you using default element names. Go to step 12.

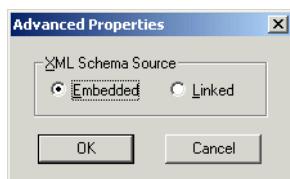
- 9 Click the Load button to load the XML schema.

The AR System will verify the XML schema and load it into the memory. You might see a message informing you that your existing XML elements and mappings will be lost, or a “schema imported successfully” message.

- 10 Click OK.

- 11 Click the Options button to display the Advanced Properties dialog box.

Figure 6-10: Advanced properties dialog box



- a Choose Embedded if you had entered a local file system path for your XSD. In this case, 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.
- b Choose Linked if you had entered a network accessible path (http or ftp) for the XSD. In this case, AR System does not store the XSD files. Neither does the WSDL 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 kind of WSDLs.

In the case of a system generated schema, it is always embedded in the WSDL.

- 12** Type a label for the web service in the Label field.

Label names can be as many as 80 characters, including spaces. This is how the web service will be displayed in the Web Services list in the server window. If there is no label, the web service name is used.

- 13** Enter an optional description in the Description field. Describe what the web service does and how it can be used, so that callers of the web service can determine if the web service has the functionality they need.

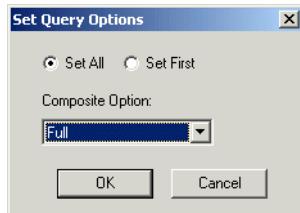
The default operations are displayed in the Operations List. The default operation types are Create, Set, and Get; the operation names are OpCreate, OpSet, OpGet, and OpGetList (there are two operations of the Get type). The same operation can be added multiple times.

- 14** Select an operation from the list.

Each operation is defined by its name and type of operation, and the Name and Type fields are automatically filled in.

For the Set operation, click the Options button to display the Set Query Options dialog box. You can set all the fields on the form, or a partial or full composite option. See “Set operations with line items” on page 388 for situations where this is applicable.

Figure 6-11: Set Query Options dialog box



- 15** Customize your operations as required.

To create a new operation:

Creating a new operation resets the mappings to the defaults.

- a** Select an existing operation from the list.
 - b** Click New.
- “Operation” will appear in the Operations List and in the Name field.
- c** Enter the new name in the name field.

- d Click Modify.

The new operation will appear in the Operations List.

To copy an operation:

Copying an operation retains the existing mapping information.

- a Select an existing operation from the list.
- b Click Copy.
- c Change the name in the name field.
- d Click Modify.

The copied operation will appear in the Operations List.

To delete an operation:

- a Select the operation from the Operation List.
- b Click Remove.

The operation will be removed from the list.

To change the name of an operation:

- a Select the operation from the Operation List.
- b Type the new name in the Name field.
- c Click Modify.

The operation with its new name will appear in the Operations List.

- 16** Enter a qualification in the Qualification bar (optional). When you select the Set or the Get operation, the Qualification bar will be enabled. You cannot use an attachment field as a field reference in a qualification.

- For the Set operation, you can enter a query that will enable 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 will identify the field whose value you want to pass back to the web service as the output parameter.

If you select a GetList operation, the Start Record and Max Limit fields contain the appropriate paths. For more information about these fields, see “Setting the starting record and setting the maximum limit” on page 381.

17 Map your parameters.

For mapping to simple and complex documents, see “Mapping to simple and complex documents” on page 384.

18 Set the Permissions, Change History, and Help Text as necessary. The permissions are the same whether the web service is visible or hidden.

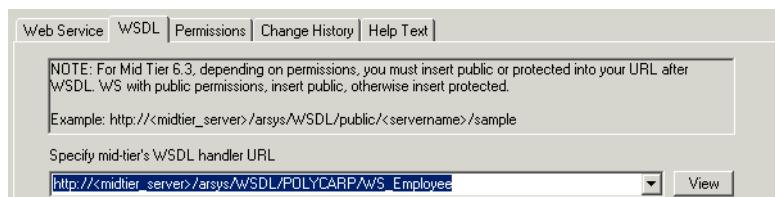
Set the permissions to Public. This is very important if you are going to publish your web service over an internet or intranet for general use.

19 Save your web service.

20 Click the WSDL tab to see the URL for your WSDL file.

A sample URL for your WSDL file is displayed in the field.

Figure 6-12: Sample URL displayed in WSDL field



21 Complete the URL as appropriate for your configuration, as follows:

- a** Replace *<midtier_server>* with the name of the web server where the mid tier is running.
- b** After WSDL, add /public or /protected depending on the web service’s permissions.

For example, if the mid tier server is POLYCARP and the web service has public permissions, use:

`http://POLYCARP/arsys/WSDL/public/POLYCARP/WS_Employee.`

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

`http://POLYCARP/arsys/WSDL/protected/POLYCARP/WS_Employee.`

22 Click the View button to view your WSDL file in the window. See Figure 6-7 on page 91 for an example.

Saving your web service

Once you have selected the base form, defined your operations, and created input and output mappings for each operation, you need to save the web service. When you save the web service in BMC Remedy Administrator, it creates a WSDL file that describes the web service. This file is in standard, formal XML format; it contains all the details necessary to interact with the service, including message formats, transport protocols, and location.

The WSDL file can be accessed with a URL using the following syntax. If the web service has *public* permissions, use:

```
http://<midtier_server>/arsys/WSDL/public/<server_name>/<webservice>
```

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

```
http://<midtier_server>/arsys/WSDL/protected/<server_name>/<webservice>
```

Once you save your web service, there are no additional steps. Your web service is ready to use. You can type the WSDL URL into your browser address field to verify that your web service is ready.

Viewing a list of web services

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

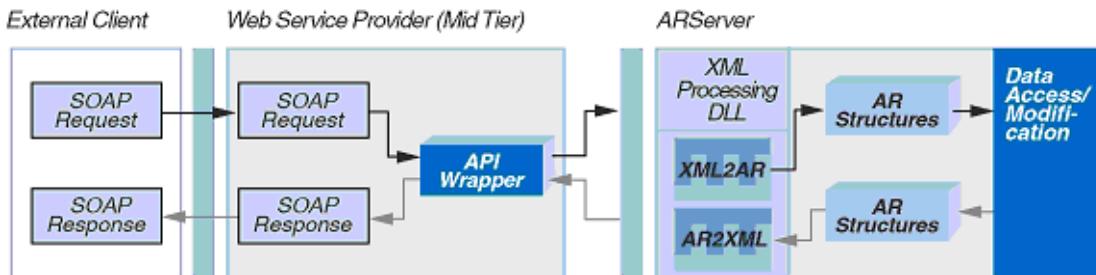
```
http://<midtier_server>/arsys/WSDL/protected/list
```

Publishing a web service

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

- Step 1** The external client sends a Simple Object Access Protocol (SOAP) request to the mid tier.

Figure 6-13: External web service client calling to AR System (Publishing)



- Step 2** The mid tier extracts the web service name and the operation name 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 (Create, Get, and Set), 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 data is treated differently according to the operation type:

- For the Get operation types, the AR System server ignores the input fields.
- For the Create operation type, the AR System server creates a new entry with the input fields.
- For the Set operation type, the AR System server searches for an entry using the expanded query string and then modifies the data using the input fields.

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 generates an XML document. The data is again treated differently according to the operation type:

- For the Get operation type, the AR System server performs a query based on the query string.
- For the Create operation type, the AR System server reads the record that was created.
- For the Set operation type, the AR System server reads the record that was modified.

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.

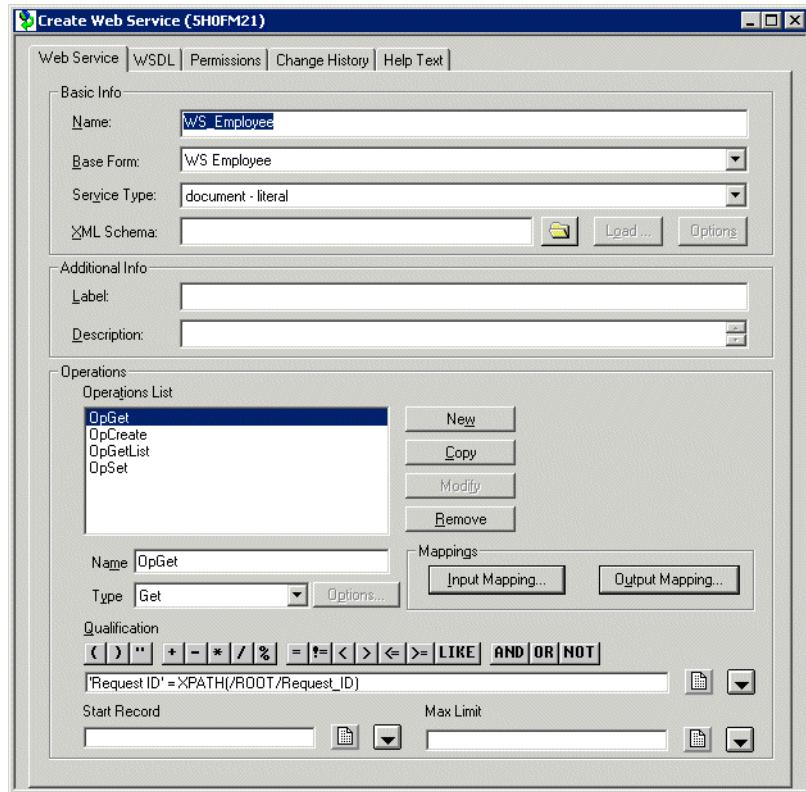
The following procedure describes how to publish a basic web service.

► **To publish a basic AR system web service**

- 1 In BMC Remedy Administrator, find the form that you want to publish as a web service in the forms list of a server window.
- 2 Right-click on the form name, and choose Create Web Service.

The Create Web Service dialog box appears.

Figure 6-14: Create web service dialog box

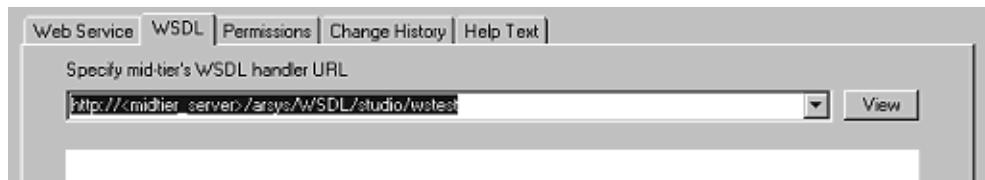


3 Qualifications are automatically created for the Get, Set, and GetList operations. You do not need a qualification for the Create operation.

The default XML type, and input and output mappings are automatically set for each operation.

4 Set the permissions to Public. This is important if you are going to publish your web service for general use on the Internet or an intranet.

Figure 6-15: URL Displayed in WSDL field



- 5 Save your web service.
- 6 Click the WSDL tab to see the URL for your Web Service Description Language (WSDL) file.

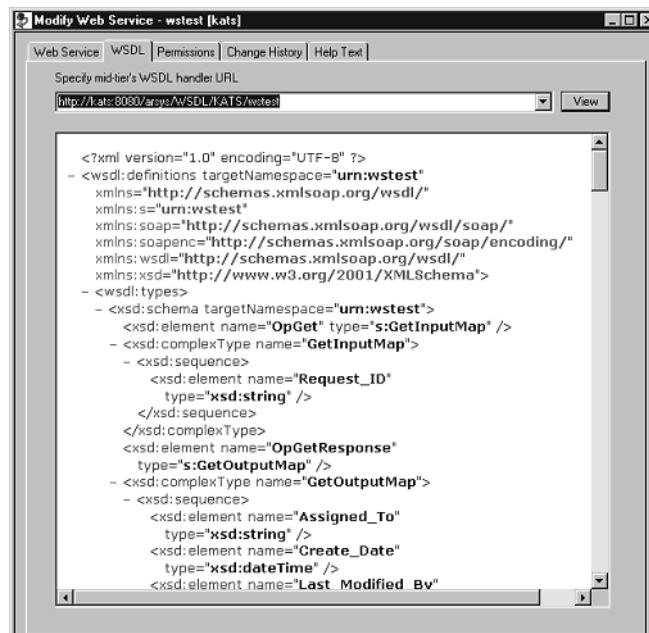
The URL for your WSDL file appears in the field.

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

In Figure 6-15, studio appears in the URL as the AR System server name. In Figure 6-16, the mid tier server is kats, the port number is 8080, and the AR System server name is KATS.

- 8 Click the View button to view your WSDL file in the window.

Figure 6-16: A WSDL file displayed in the WSDL tab of the Web Service dialog box



- 9 Use the WSDL URL or WSDL document to enable the client of your choice to consume your AR System web service.

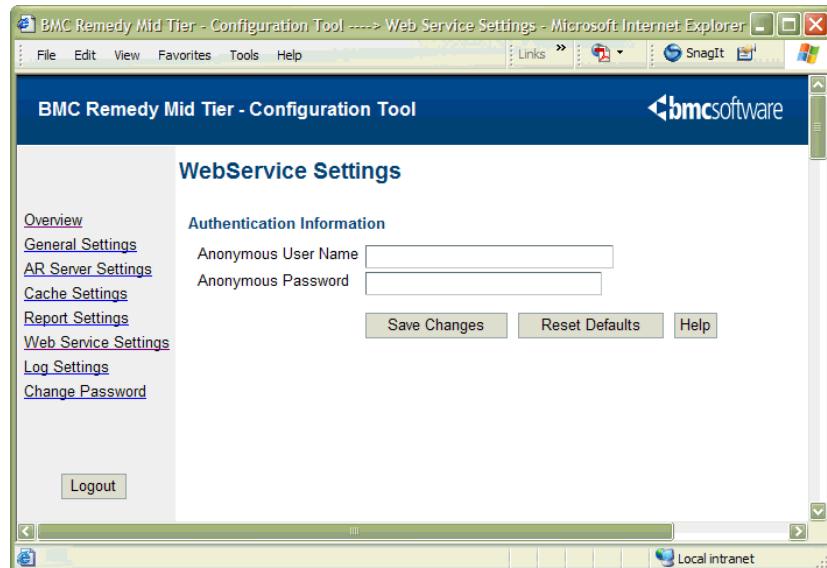
SOAP headers and authentication

When you publish a web service, BMC Remedy Administrator automatically includes a SOAP header with each operation. The web service client that calls your web service might provide a SOAP header along with the request. The AuthenticationInfo consists of userName, password, authentication, locale and timeZone elements. This timeZone element is an optional element. The SOAP header should look like this:

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

It is through the SOAP headers that the web service client can specify which user is authorized to perform the web service operations. You should not send this SOAP header unless you send it over https. If the authentication header is not specified, the user name is picked up from the BMC Remedy Mid Tier Configuration Tool.

Figure 6-17: BMC Remedy Mid Tier Configuration Tool web service settings



You can use the name Demo for development purposes, but create another user name for production, such as a guest user with a blank password.

When you use an external web service that requires a SOAP header, BMC Remedy Administrator will automatically make an element called SOAPHeader under the ROOT element. You can map elements inside the SOAPHeader just as you would regular elements. The external web service might be using the SOAP header for some other reasons not related to authentication, for example, license keys.

Note: When consuming a web service (external or AR System) using an escalation, using the SOAPHeader is required. BMC Remedy Administrator will create the SOAPHeader element under the ROOT element. You must map elements inside the SOAPHeader such as the username and password, just as you would any regular elements, to be able to communicate with the web service that needs authentication.

If you are consuming an AR System web service, BMC Remedy Administrator will create the SOAPHeader element, and it will also automatically map the user name and password to the current user name and password. It does so by setting the `arUserId=true` and `arPassword=true` attributes. If you want to pass the current user name and password to an external web service, you can do so by setting these attributes yourself. If you do not want to send the current user name and password to an AR System Web service, you must set these attributes to false.

Consuming web services

AR System 5.1 or later enables you to create applications that find and consume web services. You use a Set Fields filter action that invokes the web service using its WSDL. This Set Fields action moves the data from the web service into your AR System application. Some example web services include stock quotes and currency exchange rates.

AR System communicates with the web service through the Simple Object Access Protocol (SOAP), involving the third-party web service server (Apache AXIS) installed with the BMC Remedy Mid Tier.

Figure 6-18: Consuming an External Web Service with AR System

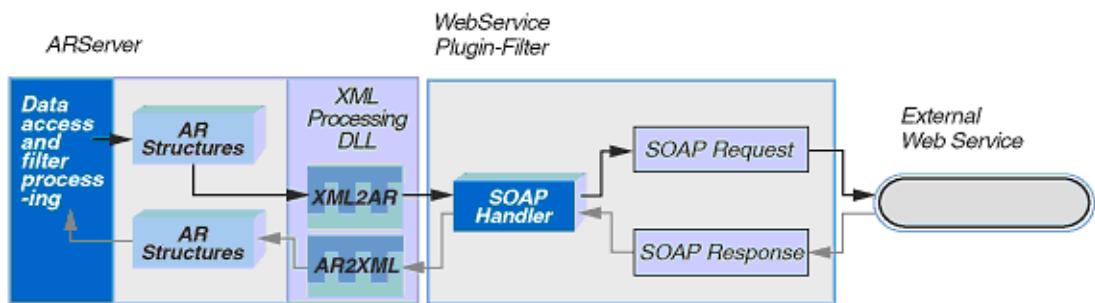
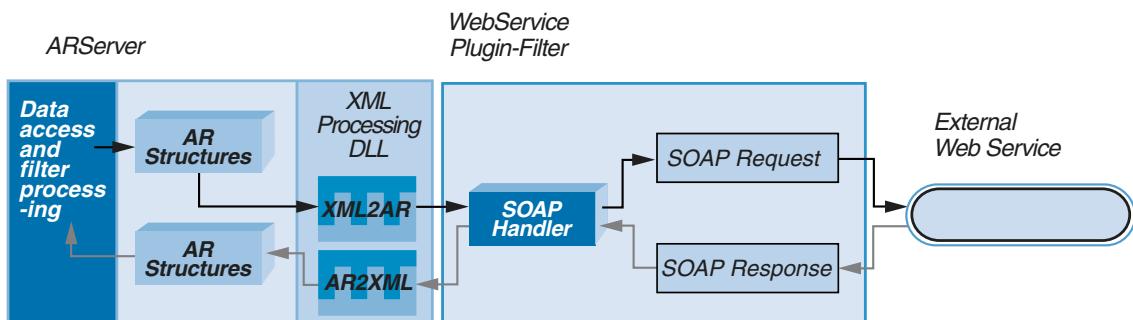


Figure 6-19: Calling out to an external web service (Consuming)



Flow for consuming a web service

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

- Step 1** A filter process triggers a Set Fields action from Web Service filter.
- 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 a WebService PlugIn Filter.
- Step 4** The WebService PlugIn Filter receives the XML document and packages it into a Simple Object Access Protocol (SOAP) request packet and calls the External Web Service.
- Step 5** The External Web Service replies back with the SOAP response packet.
- Step 6** The WebService PlugIn Filter 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 then uses the mapping information to parse the XML document and push data into the current record and into child forms (if any).
- Step 8** The Set Fields from Web Service filter action is completed.

Creating the set fields filter to import an external web service

You use an external web service by creating a Web Service Set Fields filter action to enter the data from the web service into your base form. You can then view the form in an AR System client. Make sure that you do not have other filters acting on the same form that might skew the data, or that might prevent the data from the external web service from displaying.

► To create a Set Fields filter from web service filter action

- 1 Use BMC Remedy Administrator to create a form to use as your base form for the external web service.

Before you create your fields to hold the data, check the WSDL file to see the element types that you are going to map to your fields. You can only map fields and XML elements of the same type. If you hover the cursor over the field name or XML element displayed in the mapping dialog box, the tooltip will show the data type.

- 2 In the BMC Remedy Administrator server window, select the relevant server.
- 3 Choose File > New Server Object.

The New Server Object dialog box appears.

- 4 Select Filter from the list, and click OK.

The Create Filter dialog box appears with the Basic tab selected.

- 5 Enter a name for your web services filter in the Name field.
- 6 Select or clear the Enable check box to enable or disable the filter.

You might want to disable it during development or when you are diagnosing a problem.

- 7 Enter the execution order in the Execution Order field for the filters.

The value you enter in the Execution Order field determines the order it will execute relative to others with the same triggering condition. Numbers between 0 and 1000 are valid execution order values; the lower numbers are processed first. Bear in mind that some filter actions might be in a queue and performed at a later time.

- 8 Select the Base form from the Form Name list.

This is the form to which the data will be set. You can push the data to other forms by creating workflow.

- 9 Select from the Execute On category the operation that will activate the set fields filter.

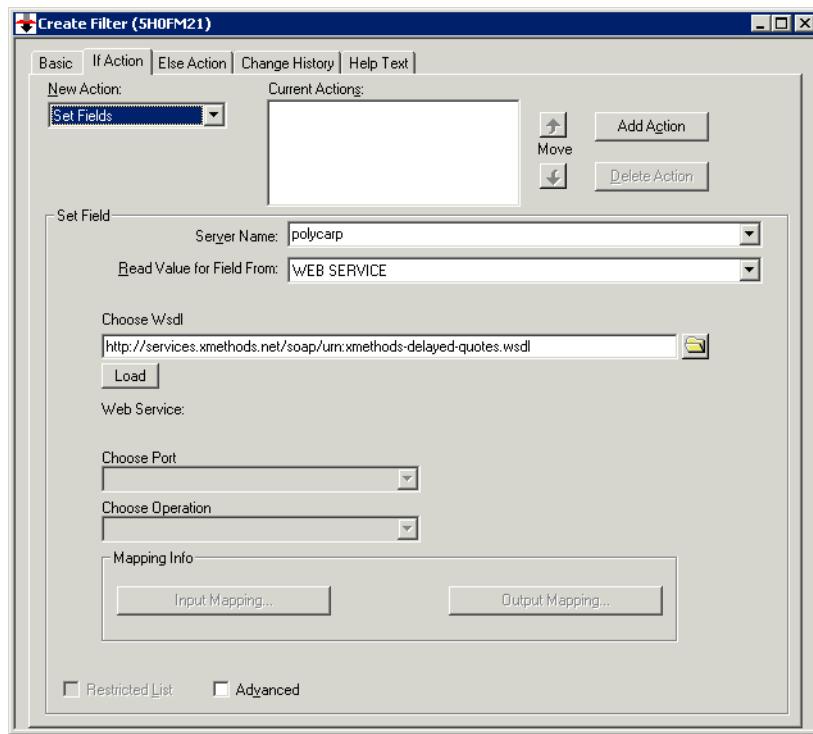
If you select multiple options, the filter will execute when any one of the selected operations occurs.

- 10 Enter a qualification statement in the Run If field if you want to refine the selection criteria.

You can type the qualification or you can build it by using the qualification bar and field list. When the qualification is met, the If Actions are executed. When the qualification is not met, the Else Actions are executed.

- 11 Click the If Action tab or the Else Action tab.
12 Select Set Fields in the New Action list.

Figure 6-20: Create Filter window



- 13 From the Server Name field list, select the server on which the web service mappings will be stored as a server object.
14 From the Read Value for Field From list, select Web Service.
15 In the Choose WSDL field, enter the URL for the WSDL file for the web service that you require.

16 Click Load.

The name of the web service is displayed automatically in the Web Service Name field.

BMC Remedy Administrator parses the WSDL file, identifies viable operations, and lists them in the Choose Operation field.



17 Select the web service port that you want to choose the operation from.

18 Select the operation that you want the web service to perform.

19 In the Create Web Service dialog box, create the input and output mapping. For mapping procedures, see “Mapping to simple and complex documents” on page 384.

20 Save your Set Fields from Web Service filter action.

You can view the Web Service data in an AR System client by opening the base form and other forms to which the data is pushed.

Consuming a web service published on the same AR System server

To consume a web service created on the same AR System server, you must have an AR System server license. You must also increase the number of threads using BMC Remedy Administrator. To do so, open the Server Information dialog box, select the Server Ports and Queues tab, and make the necessary modifications.

Threads in the Server

If an AR System server calls itself through a web service, twice the number of fast/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 be using the web services feature, if they are consuming on the same server.

Note: It is very easy to get into a deadlock situation by running out of threads because each web service call will consume two threads.

Threads in the Plug-in

The web service will use only one thread by default. If you are using multiple web services, either external or AR System, configure the plug-in with multiple threads. The number of plug-in server threads for the filter API (which is used in webservices.dll) can be specified using the `Plugin-Filter-API-Threads: <min_threads> <max_threads>` entry in the ar.cfg file. The number of threads should be configured according to the load; set the minimum number of threads initially to five. If the plugin-server is not responding in time, then increase the minimum threads.

Timeout

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

Log File

Set the `Plugin-Log-Level:100` to log to a file specified in the Server Information dialog box, Log files tab in BMC Remedy Administrator.

Web services limitations

The following section describes limitations on how web services are implemented with the AR System. Application developers should read this section carefully so they do *not* try the things mentioned in this section, but look for creative alternatives while designing their applications for web services.

XML schema limitations

Only the most common XML schema constructs are supported.

Supported XML schema constructs

The following XML schema constructs are supported in the 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 the AR System would have 20 or less elements. This element must be of a complexType.
- Namespaces
 - targetNamespace
 - chameleon namespaces
- Multiple documents
 - include
 - import
- Miscellaneous
 - annotation

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 (these are treated as strings, ignoring any restrictions)
- Name (is not supported)
- IDREFS, ENTITIES, NMOKENS (not supported)
- key, unique, keyref (these are ignored)

Note: There is no XML validation performed in the AR System.

WSDL limitations for consumption

Although most WSDLs are accepted while consuming, there are some that cause problems:

- SOAP-encoded arrays and SOAP-encoded structures are not supported. This means that rpc/encoded and doc/encoded web services with complex input or output parameters will not work. 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 are ignored.
- Overloaded operations are not allowed.

- Both input and output should be present; one-way messaging is not allowed.
- A WSDL cannot have both a `<wsdl:include>` and a `<wsdl:types>`. (As a workaround, use `<xsd:include>` inside `<wsdl:types>`. There is no restriction on the number of `<xsd:include>` that you can use.)
- A WSDL cannot have more than one `<wsdl:include>`.
- `<xsd:include>` and `<xsd:import>` inside the types sections of a WSDL cannot use a relative URL.

Chapter

7 Plug-ins

AR System plug-in services let you extend functionality beyond the AR System database. This section discusses each type of AR System plug-in and its application, potential issues surround each type, and some general information about plug-ins.

The following topics are provided:

- Overview of AR System plug-ins (page 116)
- AR system external authentication (AREA) Plug-Ins (page 122)
- Filter Plug-Ins (page 126)
- ARDBC Plug-Ins (AR System database connectivity) and vendor forms (page 127)
- Logging options (page 136)
- Plug-in aliases (page 137)
- Plug-in port numbers (page 138)

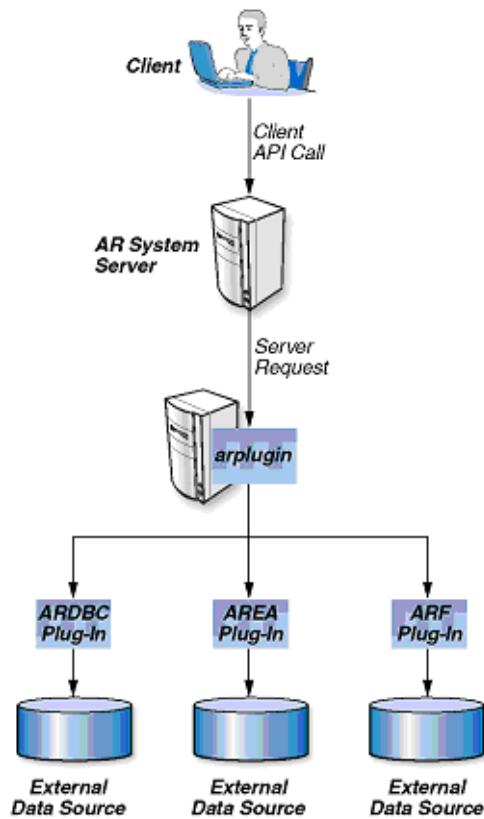
Overview of AR System plug-ins

AR System offers a plug-in service (`arplugin`) that extends its functionality to external data (data that is not contained in the AR System database).

The plug-in service supports three types of plug-ins with corresponding application program interfaces (APIs):

- Action Request System External Authentication (AREA) plug-in, which accesses network directory services or other authentication services to verify user login name and password. When you use the 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 and user passwords from external sources.
- AR System filter (ARF) plug-in, which offers an alternate method to send information requests to and from external servers. In previous versions of AR System, run processes performed external information requests. ARF uses fewer system resources than run processes use and enables the AR System server to return to its workflow faster. ARF plug-ins can also be used in escalations.
- Action Request Database Connectivity (ARDBC) plug-in, which accesses external sources of data and enables you to manipulate the data. You can integrate ARDBC with external data sources through their own APIs. The ARDBC plug-in, which you access through a vendor form, enables you to perform the following tasks:
 - Search external data sources.
 - Create, delete, modify, and set entries.
 - Populate search-style character menus.
 - Implement workflow.

Figure 7-1: AR System Plug-In architecture



Creating plug-ins

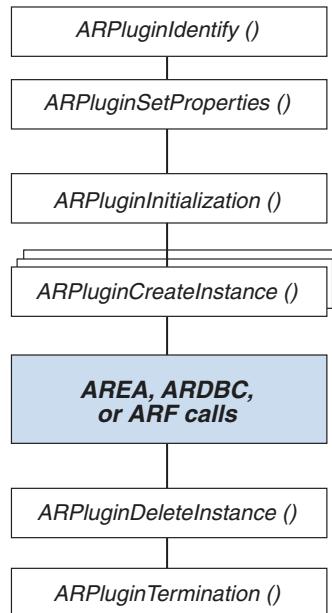
You use the AR System C API to extend *client* functionality. To use the C API, you write code that references the C API and a dynamically linked library (.dll) if you use a Windows platform, or a shared object library if you use a UNIX platform. AR System provides the C API, header files, and libraries, which are installed into the same directory as other AR System files. You use AR System plug-ins to extend *server* functionality. To use AR System plug-ins, you must create a .dll or shared object that conforms to the specifications in the header files. BMC Remedy provides files that contain empty function definitions that you can use to build your plug-in.

Note: Creating an AR System plug-in requires that you install the API package when you installed AR System.

► To create and implement a plug-in

- 1 Write a C or C++ program that includes:
 - Plug-in API calls for initialization, termination, object creation, and object deletion
 - Plug-in type-specific API calls—AREA, AR Filter, or ARDBC
 - Code to implement the calls
- 2 Configure your system to recognize your new plug-in. For more information, see the *Configuring* guide.
- 3 Place your plug-in in the directory that contains the other AR System files. At run time, the plug-in server reads the configuration file and loads the specified plug-ins.

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

Figure 7-2: Structure of an AR System plug-in

- 4 Use sample Microsoft DevStudio projects (Windows) or makefiles (UNIX), which you install with AR System, to build your program into your .dll or shared object library.
- 5 Run the `arplugin` or `arplugin.exe` program from the command line.

The `arplugin` program reads the configuration file at runtime and loads the plug-ins that the configuration file specifies.

Note: If you do not add the plug-in information to the configuration file before you run `arplugin`, the plug-in server will not load any plug-ins. If you consequently modify the configuration file, you must restart the plug-in server to load the plug-ins.

Common plug-in API functions

This section describes common plug-in API functions. See the *C API Reference* guide for more information.

In general, all these plug-in callback functions use the same structure as other AR System API functions. The plug-in server calls these 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 service.

WARNING: Plug-in operations run synchronously; that is, AR System waits for a plug-in to complete its processing before the server will continue its processing. Thus, a badly written plug-in can adversely impact AR System performance and stability. A plug-in developer must pay particular attention to the following areas: thread safety, minimal processing logic for optimal code execution, only implement callback methods that are required or have custom logic for a given plug-in, allocate/free resources, as appropriate, to prevent resource leaks and, finally, optimize any costly operation that must be performed (or data structures that must be built) by employing a meaningful caching strategy.

Plug-in API	Description
ARPluginIdentify	Returns plug-in identity information to the plug-in server, including plug-in type, name, and version. At runtime, this is the information used by the system to see the plug-in. The plug-in server calls this function once, when initially loading the plug-in.
ARPluginSetProperties	(Optional) Provides information to the plug-in, such as plug-in service configuration parameters or services that the plug-in server provides to the plug-ins it loads.
ARPluginEvent	(Optional) Notifies the plug-in that events have occurred. The plug-in can use such notification to trigger a reaction to changes in configuration.
ARPluginInitialization	(Optional) Allocates and initializes global resources when the arplugin server loads the plug-in. The plug-in can use these resources for the life of the plug-in's instance.

Plug-in API	Description
ARPluginCreateInstance	(Optional) Creates a plug-in instance and instance-specific data. This function has a multi-threaded architecture, which allows the plug-in server to support better scalability/throughput capabilities.
ARPluginDeleteInstance	(Optional) When a thread encounters an exception, the plug-in server uses this call to free resources associated with the instance.
ARPluginTermination	(Optional) Deallocates global resources for a plug-in instance when the plug-in instance completes its function.

Running the arplugin server

Normally, the arplugin server will be started automatically when the AR System server is started. To run the arplugin server manually, type the following at the command prompt:

- **UNIX:** arplugin [-i <directory>] [-s]
- **Windows:** arplugin [-i <directory>] [-m] [-s]

Option	Operating system	Description
-i <directory>	UNIX and Windows	Specifies path to the AR System server installation directory.
-m	Windows	Specifies manual running mode for the arplugin server. The default mode is to run the arplugin server as a Windows NT service.
-s	UNIX and Windows	Specifies AR System server name.

Accessing the plug-ins

After you have completed the preparatory steps, your plug-ins are ready to be accessed under the following circumstances:

- AREA plug-ins are accessed when the AR System server needs to authenticate the identity of a user. The AR System server uses AREA plug-ins in a manner that is transparent to the user. Plug-ins connect to the AR System server through the arplugin server.
- ARF plug-ins are accessed when server-side workflow, such as filters and escalations, reference filter API calls.
- ARDBC plug-ins are accessed when an API client or workflow operation references data stored in an external data source.

Issues and considerations

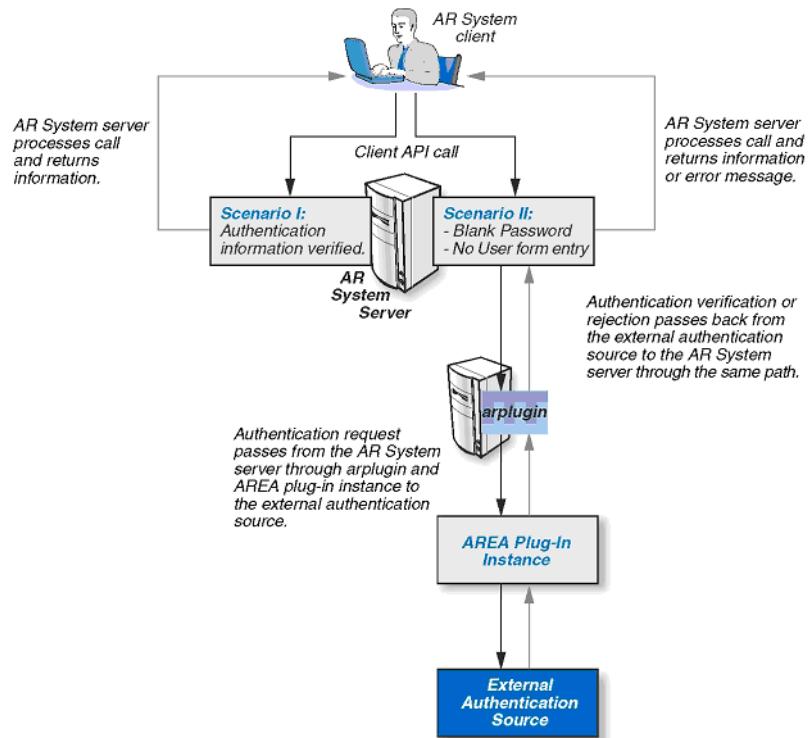
You must create a separate plug-in for each of the three types of plug-ins. For example, you cannot create one plug-in that supports both AREA and ARDBC.

AR system external authentication (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. For example, data sources such as LDAP and Kerberos, which can be accessed by most network applications, are supported.

When you install AR System, you have the option to install a sample AREA LDAP implementation, including an AREA LDAP plug-in. This plug-in provides you with an integration point between AR System and LDAP directory services. To integrate AR System with an external authentication service other than LDAP, you must create your own plug-in. See “Creating plug-ins” on page 118 and “AREA plug-in specific API functions” on page 124.

Figure 7-3: Data Flow of an Example AREA Login Request

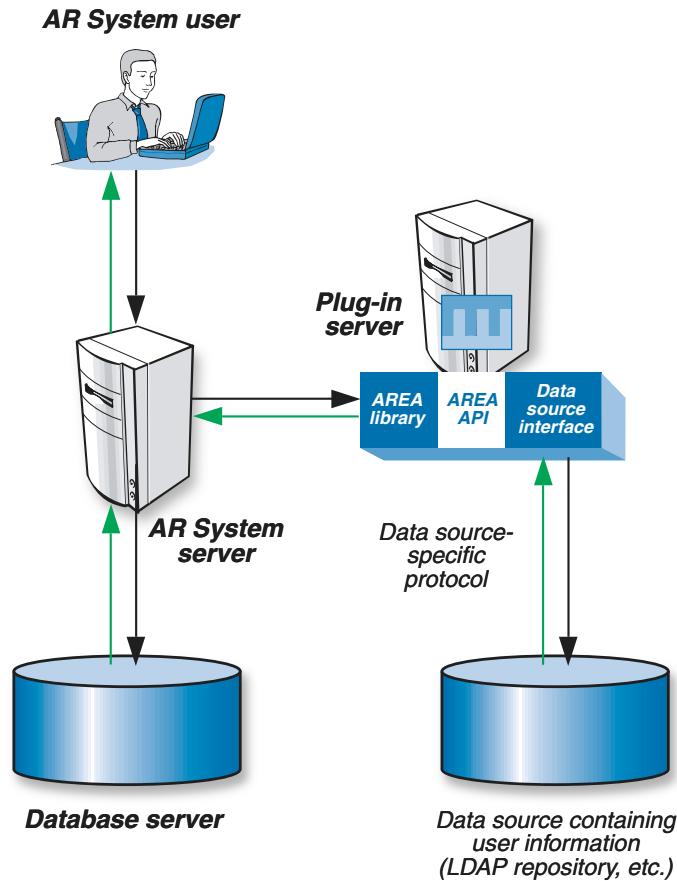


AREA plug-in specific API functions

When you install AR System, you have the option to install sample AREA API implementations—one that allows a single user to log in and use a floating license, and one that uses the Netscape LDAP SDK. This section briefly describes AREA plug-in-specific API functions.

AREA plug-in API	Description
AREAFreeCallback	(Optional) Frees the data associated with the AREAResponseStruct, which the AREAVerifyLoginCallback call returns. Although the arplugin server normally frees the storage that other API calls return, it does not free the storage that this API call returns. Instead, the plug-in de-allocates the storage through this call, allowing the plug-in to cache information optionally.
AREAVerifyLoginCallback	The arplugin server issues this call when the AR System server makes a request to authenticate a user. The AR System server passes the unencrypted user name and password as parameters. You must protect any global information or resources accessed here with appropriate mutual exclusion locks to ensure multithread safety.
AREANeedToSyncCallback	Determines whether the user information in the server cache is invalid. The AR System server periodically checks with the AREA plug-in to determine if any in-memory copies of user information have expired. The arplugin server issues this call if it receives an authentication request from the server at least five minutes since the last authentication request.

Figure 7-4: External authentication architecture



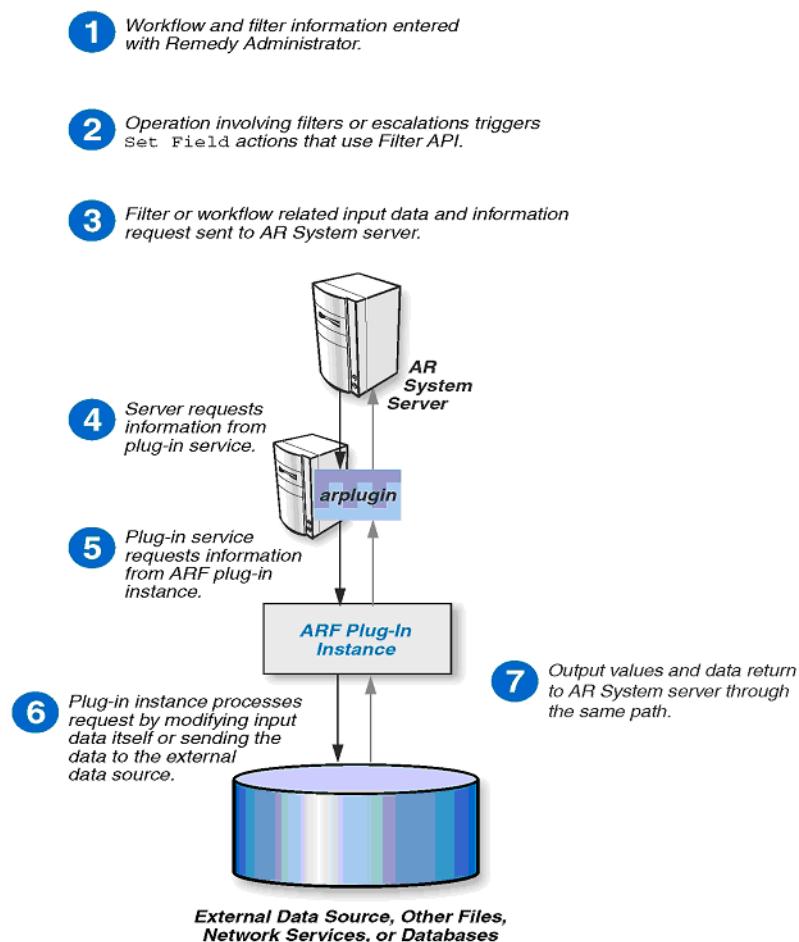
Issues and considerations

- For the AR System server to use the AREA plug-in to authorize logins, those users' entries in the User form should have blank passwords.
- For more information about the AREA API, see the *C API Reference* guide.

Filter Plug-Ins

Filter plug-ins allow 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. Since this middleware is loaded as a plug-in when AR System is started, instead of at each event as a standalone executable, it consumes fewer resources and less processor time than a Set Fields \$PROCESS\$ action.

Figure 7-5: Data flow of a filter plug-in request



Filter Plug-In specific API function

The arplugin server calls the ARFilterApiCall function when it receives filter API set fields actions data or input values from the AR System server during workflow operations. Input parameters and output values are both passed as an array of type ARValueList. The variables in both arrays are defined by you according to the needs of your integration.

Issues and considerations

Unlike AREA plug-ins, the arplugin server can load multiple filter plug-ins simultaneously.

The filter plug-in function is a blocking call, so the AR System server thread that makes the call will wait for your plug-in to respond. For best performance, your plug-in should return quickly. You should provide instructions to those who will install your plug-in regarding the expected latency and have them set their AR_SERVER_INFO_FILTER_TIMEOUT value accordingly.

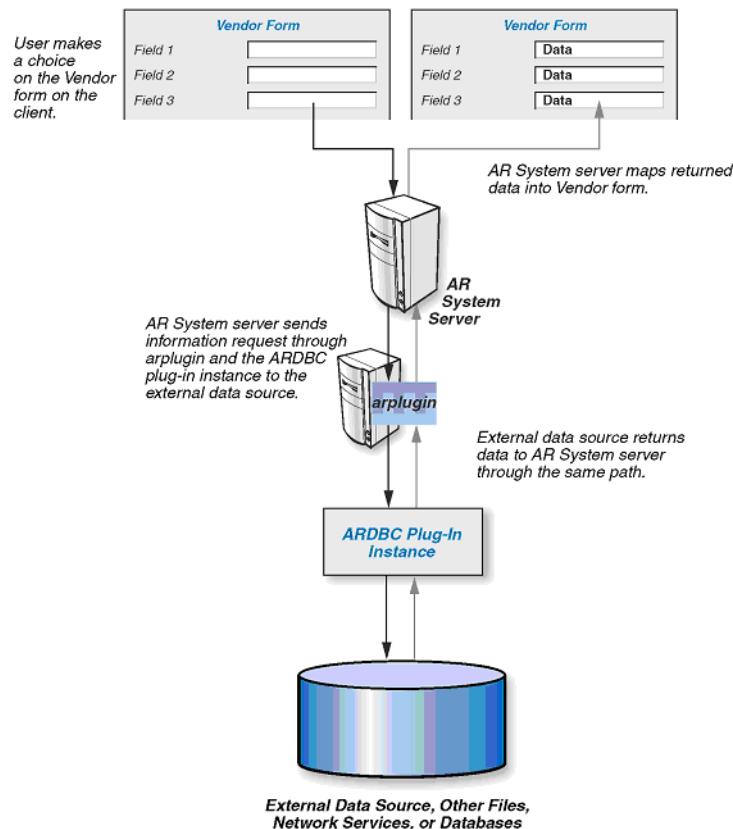
For more specifics on writing a filter plug-in, see the *C API* guide.

ARDBC Plug-Ins (AR System database connectivity) and vendor forms

ARDBC plug-ins allow local-level access to external data sources from AR System. Once an ARDBC plug-in has been installed, the AR System administrator can create a new vendor form that references the table and columns of the outside data source. Then views and workflow can be built for vendor forms just as if they were regular AR System forms. The source of data manipulated by BMC Remedy User will be transparent to the operator.

When users enter requests in the vendor form on AR System API clients, such as BMC Remedy User or Mid Tier, the AR System server sends the requests to the arplugin 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.

For more information about vendor forms, see Chapter 11, “Vendor forms.”

Figure 7-6: Data Flow of an ARDBC Plug-In Request

ARDBC Plug-In specific API functions

In addition to the standard AR plug-in functions described previously, building an ARDBC plug-in requires you to implement calls on your external data source that are analogous to set entry, get entry, create entry, delete entry, and get list C API calls.

When you implement an ARDBC plug-in, be sure to completely document the capabilities of your plug-in. For example, you might implement 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, allow the default behavior to occur. For example, if your data source does not support transactions, do not define ARDBCCommitTransaction and ARDBCRollbackTransaction calls.

The following table describes the ARDBC plug-in-specific APIs.

ARDBC plug-in API	Description
ARDBCCommitTransaction	(Optional) Commits the changes of one or more previous ARDBC calls to the external data source. The client typically calls this function after it performs one or more actions that read or modify the external data source. After the client issues this call, the AR System server begins the next transaction.
	Note: If you do not define this function and the arplugin server receives a commit transaction request, the call proceeds without error.
ARDBCCreateEntry	Creates an entry in the external data source. The arplugin server issues this call upon request from the AR System server.
	Note: If you do not define this function, this call does not create an entry and the AR System server will receive an error message from the arplugin server.
ARDBCDeleteEntry	Deletes a single entry in an external data source. The arplugin server issues an ARDBCDeleteEntry call when it receives a delete entry request from the AR System server.
	Note: If you do not define this function, the AR System server will receive an error message and will not delete the entry.
ARDBCGetEntry	Retrieves a single entry from an external data source table. The arplugin server issues this call when the AR System server receives an ARGetEntry request from a client.

ARDBC plug-in API	Description
ARDBCGetEntryBLOB	<p>Opens an attachment. The AR System server receives an ARGetEntryBLOB API call from the client when users request to open an attachment from an external form. The server then sends a request to the arplugin server, which issues an ARDBCGetEntryBLOB call.</p> <p>Note: If you do not define this function and the arplugin server receives a get entry BLOB request, the AR System server will receive an error message and will not retrieve the data.</p>
ARDBCGetEntryStatistics	<p>Gathers statistical information about a set of data. The arplugin server issues this call when it receives a request from the AR System server to get entry statistics.</p> <p>Note: If you do not define this function and the arplugin server receives a get entry statistics request, the AR System server will receive an error message and will not retrieve the data.</p>
ARDBCGetListEntryWithFields	<p>Retrieves a list of qualified entries with field data. The arplugin server issues this call when the AR System server receives an ARGetListEntryWithFields or an ARGetListEntry request from a client.</p> <p>Note: If you do not define this function and the arplugin server receives a get list entry with fields request, the AR System server returns a list of zero entries.</p>
ARDBCGetListSchemas	<p>Retrieves a list of external schemas. The arplugin server issues this call when the AR System server receives an ARGetListExtSchemaCandidates request from the client to list candidate schemas. In ARPluginIdentify, you specify the plug-in name in the ID structure. For the call to ARDBCGetListSchemas to complete successfully, that name must match the name specified in schema->compoundSchema[n].u.vendor.vendorName.</p> <p>Note: If you do not define this function and the arplugin server receives a get list schemas request, the AR System server returns a list of zero forms.</p>
ARDBCGetMultipleFields	<p>Retrieves a list of fields from an external data source. The plug-in issues this call when the AR System server receives an ARGetMultipleExtFieldCandidates API call from a client. For example, a BMC Remedy Administrator client might provide a user with a list of fields instead of requiring the user to type in a field name.</p> <p>Note: If you do not define this function and the arplugin server receives a get multiple fields request, the AR System server returns a list of zero fields.</p>

ARDBC plug-in API	Description
ARDBCRollbackTransaction	<p>Rolls back the changes of one or more previous ARDBC calls to the external data source. The arplugin server issues this call when one or more actions that read or modify the external data source result in a failed operation or an error. After the arplugin server issues this function, the AR System server begins the next transaction.</p>
	<p>Note: If you do not define this function and the arplugin server receives a roll back transaction request, the AR System server does not process the request and does not return an error.</p>
ARDBCSetEntry	<p>Modifies the contents of a single entry or BLOB. If the specified entry does not exist, the plug-in creates it. An ARDBCSetEntry operation on a non-existent entry can occur when the AR System server receives an ARMergeEntry API call from a client.</p>
	<p>Note: If you do not define this function, the AR System server will receive an error message and this call will not set the entry.</p>

Calling AR system API from ARDBC plug-in

 NEW

You can make AR System API calls from the ARDBC plug-in. In previous versions of AR System, such a call would have to be made with a known user account. With AR System 7.0, you can make those calls as the same user whose operation led to the ARDBC plug-in call. This makes sure that any call from the ARDBC plug-in has the same permissions as the user who called the ARDBC plug-in in the first place.

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 server. The plug-in server provides call back routines to fetch the user name, authentication string, and GUID. Subsequently, when a plug-in wants to make an API call, it uses those call back routines to fetch the information it needs to authenticate itself as the user that made the original call to the plug-in server.

The calling plug-in uses the following API calls to set the call back routines for the API to be able to fetch user name, authentication string, and the authenticating GUID:

- ARSetUserNameSource
- ARSetAuthStringSource
- ARSetNativeAuthenticationSource

Pointers to the call back routines themselves 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 ARInitialization call, before making any other API calls.

Note: When using the GUID authentication feature from a plug-in, internal users (such as ESCALATOR and ARCHIVE) will encounter errors. The errors occur because these users are not valid users for making API calls.

See the *C API Guide* for more information.

Creating a vendor form using an ARDBC Plug-In

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 *Configuring* guide.

Note: Creating a vendor form for an ARDBC LDAP plug-in is a special case.

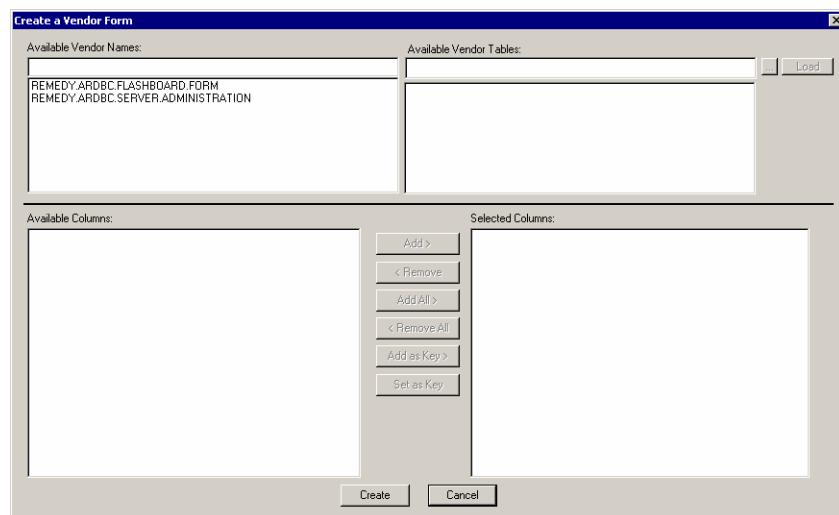
For more information, see “Creating a vendor form to represent a collection of LDAP objects” on page 146.

► To create a vendor form using an ARDBC plug-in

- 1 From the New Server Object dialog box in BMC Remedy Administrator, select Vendor Form.

The Create a Vendor Form dialog box appears.

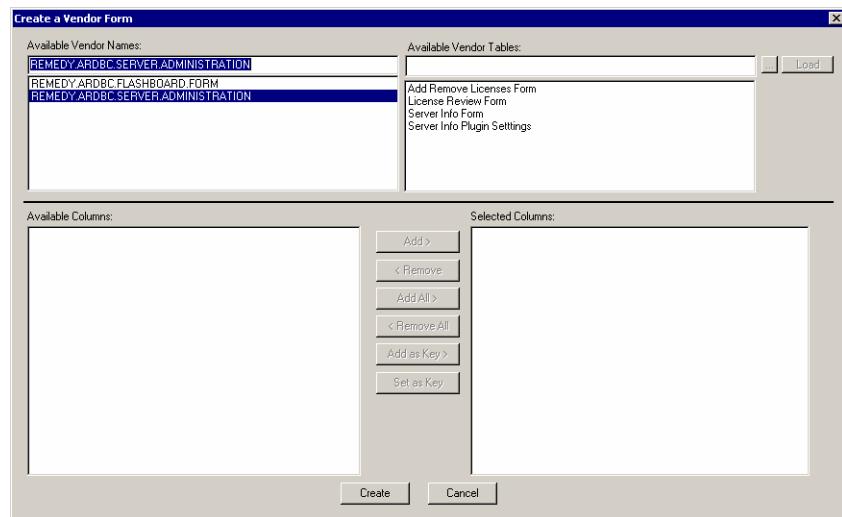
Figure 7-7: Create a vendor form dialog box



- 2 Choose the ARDBC plug-in that you want to use in the list of Available Vendor Names.

The names of the tables available from the vendor name you selected appear in the Available Vendor Tables field.

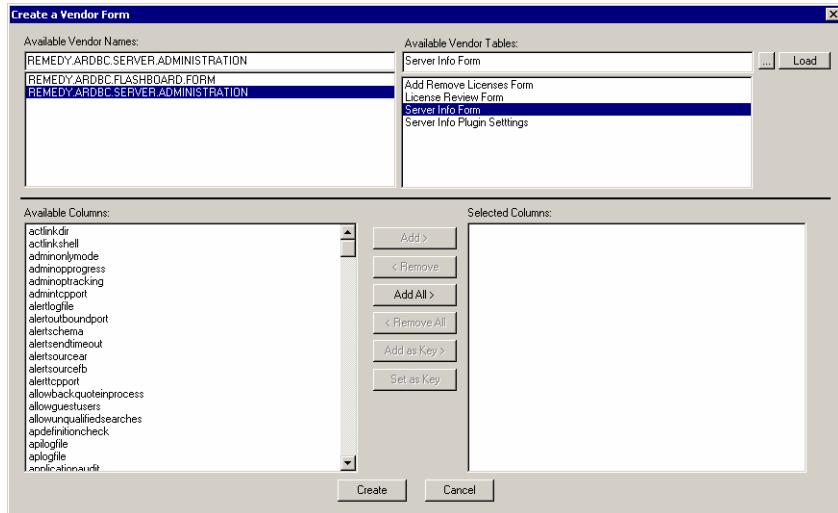
Figure 7-8: Available vendor tables field filled in



- 3 Choose the remote data table that you need to access in the list of Available Vendor Tables.

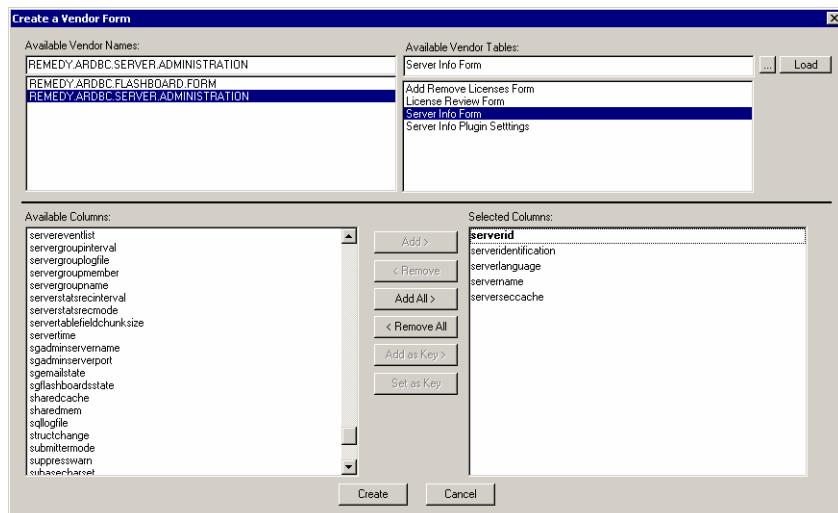
The fields in the chosen table appear in the Available Columns list.

Figure 7-9: Available columns field populated



- 4 Select the columns that you want to access in the AR System, and click Add. The columns you select appear in the Selected Columns field.

Figure 7-10: Selected columns field filled in



- 5 When you finish, click Create.

The vendor form for this plug-in and table will be created and will appear in BMC Remedy Administrator for you to edit.

Issues and considerations

Keep the following issues in mind when creating a vendor form:

- The arplugin server can load more than one ARDBC plug-in at a time.
- A vendor form cannot be used in join forms.
- FTS (Full Text Search) operations are not available on vendor form fields.
- You can only add Required and Optional fields to the Vendor form that correspond to actual columns within the data source. In addition, you can only add a Display Only field when the column name does not correspond to a column within the data source.
- Attachment fields are not supported on vendor forms.
- For more information about vendor forms, see the *Form and Application Objects* guide. For more information about the ARDBC API, see the *CAPI Reference* guide. For more information about configuring your AR System server to recognize a plug-in, see the *Configuring* guide.

Logging options

You can view information in the plug-in log file (`arplugin.log`). It can be accessed from the path

`<ar_server_install_directory>\ARserver\Db\arplugin.log` or in BMC Remedy Administrator (in the Logging section of the Server Information dialog box).

To specify the level of plug-in service logging, set a log level in the `ar.cfg` or `ar.conf` file under the setting `PLUGIN_LOG_LEVEL`. For more information about plug-in log levels, see the *Configuring* guide.

Plug-in aliases

You can define aliases for plug-ins. These aliases redirect AR System to access the actual plug-in. Before initiating a call to the plug-in server, the AR System server determines whether or not an alias has been defined for the plug-in. If an alias has been defined, the AR System server is redirected to the specified host and port number.

Defining plug-in aliases

You define plug-in aliases in the `ar.cfg` configuration file. Use the following format to define plug-in aliases:

```
Server-Plugin-Alias: alias-name real-name host-name[:port-number]
```

Parameter	Description
alias-name	The name referenced in AR System applications. AR System Filter (ARF) API calls and vendor forms reference this alias name. Note: The alias-name is an arbitrary string, but it cannot include semicolons or blank-space characters, such as spaces, tabs, or new lines.
real-name	The actual name that the plug-in exposes to the plug-in server.
host-name	The name of the host the AR System server will access to find the associated plug-in server.
port-number	The port number the AR System server will connect with when accessing the associated plug-in server. This is optional. If you do not specify a port number as part of the alias the Plugin-Port is used.

Examples of plug-in aliases

The following are examples of how aliases affect the behavior of the AR System server when accessing 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.AR.F.PLRL.myhost RMDY.AR.F.PLRL myhost

Workflow that accesses the ARF plug-in named RMDY.AR.F.PLRL.myhost is redirected to the RMDY.AR.F.PLRL 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.

Plug-in port numbers

When AR System accesses the plug-in server, it attempts to use port numbers in the following order:

- 1 The port number of the plug-in alias.
- 2 The port number specified by `Plugin-Port` setting in the `ar.cfg` configuration file.
- 3 The port number that the plug-in server has registered with the portmapper. If the plug-in server is configured to not register with the portmapper, then the AR System server will not be able to access plug-ins.

Chapter

8 LDAP plug-ins

This section describes how to configure and use the ARDBC and AREA LDAP plug-ins to integrate the AR System with a directory service. The following topics are provided:

- Overview of LDAP and AR System (page 140)
- ARDBC LDAP plug-in (page 140)
- AREA LDAP plug-in (page 152)

Overview of LDAP and AR System

Light-Weight 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, that user can log in to any application that supports the LDAP protocol with the same user name and password.

The AR System Database Connectivity (ARDBC) LDAP plug-in allows you to access data objects stored in a directory service as if they were entries stored in a typical AR System form. You can search, modify, and create data in a directory service using this plug-in. You can also use this data to participate in workflow as well as to populate character menus and table fields.

The AR External Authentication (AREA) LDAP plug-in enables you to authenticate AR System users against external LDAP directory services.

ARDBC LDAP plug-in

The ARDBC LDAP plug-in allows you to access data from an external LDAP system using 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 your external LDAP data as if it were stored in the AR System database. For more information about vendor forms, see Chapter 11, “Vendor forms.”

Requirements

Keep the following considerations in mind when using the ARDBC LDAP plug-in:

- The ARDBC LDAP plug-in issues requests to directory services using the LDAP v3 protocol.
- Attributes consist of either character data, integer data, or timestamps. Attachments are not supported.
- LDAP does not support transactions. Consequently, once an object is created, modified, or deleted, the change will not roll back should subsequent workflow in the AR System server detect an error condition.

- By default, all attributes have one value. Multi-valued attributes are supported using a special notation described later in this section.
- The distinguished name and password specified in the ARDBC LDAP configuration are used when connecting to any directory services referenced in LDAP search URLs.
- Only server-based certificates are supported.

Configuring the ARDBC LDAP plug-in

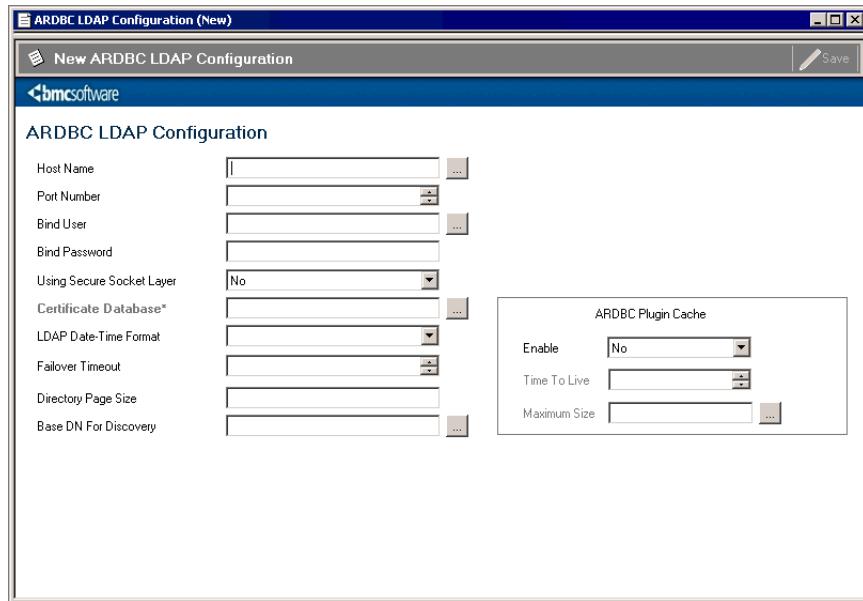
You must configure the ARDBC LDAP plug-in *before* you create the vendor form used to access user information in your particular LDAP server.

You configure ARDBC is 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 two forms, ARDBC LDAP Configuration and Configuration ARDBC. The Configuration ARDBC is a vendor form, using a separate plug-in, that reads and writes to the ar.cfg file. The ARDBC LDAP Configuration form is 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 Open BMC Remedy User and log in to the AR System server as a user in the Administrator group.
- 2 Choose File > Open > Object List to display a list of forms, guides, applications, and entry points.
- 3 Select the ARDBC LDAP Configuration form, and click New.

The ARDBC LDAP Configuration form opens.

Figure 8-1: ARDBC LDAP configuration form

- 4 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 hostnames up to 255 characters long. Starting with the first hostname in the list, AR System will attempt to connect to each server until it is successful.

If you are using Secure Socket Layer (SSL), this host name should match the name for which the server's certificate was issued.

- 5 In the Port Number field, enter a port number for this directory service. The default port number is 389, or 636 if using an SSL connection.
- 6 In the Bind User field, enter the distinguished name of the user account that the ARDBC LDAP plug-in will use when logging in to the directory service. This name was designated by the administrator who set up the LDAP service. With the vendor form, some LDAP servers will 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`.

- 7 In the Bind Password field, enter the password for this user account. (For security, asterisks replace the characters you enter for the password.)

If you leave the Bind Name and Password fields blank, you will be connected anonymously.

- 8 To use an Secure Socket Layer (SSL) connection, select Yes in the Using Secure Socket Layer field; otherwise, accept the default value of No. If you select Yes, the Certificate Database field becomes active, and you can enter a certificate database as described in step 9. Since SSL requires additional setup in this form and outside the AR System, you might first want to experiment without SSL and then add this option later.

- 9 In the Certificate Database field, enter the path to the directory containing the certificate database file. The function assumes that the database file is named cert7.db, but do not include the file name in the path. You also need to have the file key3.db present in the same directory.

You must obtain a certificate database file since the AR System uses the Mozilla LDAP SDK, which requires this file for connection over SSL. The certificate database file cert7.db is used by the client (that is, the LDAP plug-in) to determine if the certificate sent by the LDAP server can be trusted.

- 10 In the LDAP Date-Time Format field, select the format to be used to represent date and time to LDAP servers.

Value	Description	Example: 6 am, september 28, 2001
0	Generalized Time Format YYYYMMDDHHMMSSZ This format is recognized by all LDAP servers and is recommended.	20010928060000Z

Value	Description	Example: 6 am, september 28, 2001
1	AD Generalized Time Format YYYYMMDDHHMMSS.OZ This format is only recognized by Microsoft Active Directory servers.	20010928060000.0Z
2	UTC Time Format YYMMDDHHMMSSZ This is a historical format and does not indicate the century. It is not recommended.	010928060000Z

For more information about date and time formats, see the *Configuring* guide.

- 11 In the Failover Timeout field, specify the number of seconds within which the directory service must respond to the plug-in server before an error is returned. The minimum value is 0. In this case, the connection must be made immediately. The failover timeout cannot be set higher than the value of the Server-Plugin-Default-Timeout parameter.
- 12 In the Directory Page Size field, enter the number of entries to be returned in a single page to the client from the external directory server when processing a search request.

Tip: Setting the Directory Page Size to 1000 can help improve your system's performance while you are designing and creating vendor forms.

- 13 In the Base DN for Discovery field, enter a base DN to be used instead of the root DN 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 are designing and creating vendor forms.

- 14 Specify ARDBC plug-in caching information in the ARDBC Plug-in Cache box.
 - a In the Enable field, specify 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 runtime.

- 15 Click Save.

The system updates the `ar.cfg` (`ar.conf`) file with the parameters you have specified in this form.

For more information, see the *Configuring* guide.

Building AR system forms for directory services

This section describes the concepts and generic procedures involved in building vendor forms and workflow in the AR System to access data stored in a directory service. The following topics are covered in this section:

- “Organizing data”
- “Creating a vendor form to represent a collection of LDAP objects” on page 146
- “Identifying objects uniquely” on page 148
- “Supporting object creation” on page 149

Organizing data

Data within a directory service is organized differently than traditional database applications. Traditional database applications organize data within tables that have a fixed number of columns. Each row in that table represents a single entity and contains a value for each column that the table defines.

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.

Table 8-1: “Best fit” analogies between data sources

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.

Note: The example in Appendix E, “ARDDBC LDAP example: accessing inetorgperson data” walks you through creating an AR System vendor form and workflow to create and access objects that belong to the inetorgperson object class.

Creating a vendor form to represent a collection of LDAP objects

A table within 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 within a directory service is similar to entries in a form or a collection of rows in a table. When working with a directory service, a collection of objects can be described using a standard LDAP search URL. An LDAP search URL looks something like this:

```
ldap://orangina/o=remedy.com??sub?(objectclass=inetorgperson)
```

This URL contains the following components:

Table 8-2: LDAP URL components

URL component	What it means
ldap://	Indicates the LDAP protocol.
orangina	The directory service host name.
o=remedy.com	The search base name.
sub	The search scope. In this case, sub indicates that the search applies to the entire sub-tree under the base name.
(objectclass=inetorgperson)	A filter that selects the objects.
	Note: It is recommended that you 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 allows 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 within the URL. In the previous example, no attributes are listed because the LDAP plug-in ignores the attribute list. Instead, you identify attributes within the field properties—for more information, see “Alternative method of adding a field to represent the uid (User ID) attribute” on page 471.

Each object selected by the LDAP search can be represented as an entry in a vendor form. You use BMC Remedy Administrator to create a vendor form and add fields to which you attach LDAP data.

For general information about creating vendor forms, see Chapter 11, “Vendor forms.”

For an example of creating a vendor form specifically for LDAP data, see Appendix E, “ARDBC LDAP example: accessing inetorgperson data.”

Identifying objects uniquely

The AR System uniquely identifies entries within a form through the Request ID field.

Objects within a directory service define an attribute called the Distinguished Name (`dn`) that uniquely distinguishes one object from another. An object's distinguished name often consists of one or more other 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 the 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 will be used to uniquely 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 will receive a ARERR (100) Entry ID list is empty message, and no records will be displayed in the client. If more than one record has the same value, you will retrieve data only for the first matching entry.

For example, in a typical system the DN (distinguished name) 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 runtime errors or incorrect behavior.

Supporting object creation

This section describes how to create new objects in the directory service using the ARDBC LDAP plug-in. To support the creation of objects using the ARDBC LDAP plug-in, you must perform the following tasks:

- Create an AR System field that is associated with the `objectclass` attribute. Note that the `objectclass` attribute is a multi-value attribute.
- Create a field (other than Request ID) associated with `dn` and define workflow that will assign a value to it. Although entries within the AR System are uniquely identified by the Request ID, objects within 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 within the AR System.

Creating an `objectclass` field

`objectclass` is a multi-valued attribute that describes all of the 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 of the combined attributes.

When you create an object, you must specify all of the object classes from which the object inherits. You must add a character field to your form and attach this field to the `objectclass` attribute and use the multi-value attribute notation mentioned previously.

As 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` would be:

```
top, person, organizationalperson, inetorgperson
```

`inetorgperson` objects inherit from the `top`, `person`, `organizationalperson`, and `inetorgperson` object classes.

As the value does not change for this field, you should make this field Read Only. You might also want to make the field Hidden.

Multi-valued attributes

Most attributes within 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 zero, one, two, or any number of telephone numbers as atomic values.

This differs from typical database applications and the AR System in that a column or field only stores one value. If you want to store two phone numbers in such an application, you would 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.

```
<attribute name>[*<separator string>]
```

Values associated with `<attribute name>` are concatenated into a single value in the AR System but separated with `<separator string>`. 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

Distinguished Name (`dn`) is an attribute that is assigned a value generally through workflow. The workflow will take one or more values and assemble the value for the `dn` attribute. Once 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 how to create a filter, see the *Workflow* guide.

ARDBC LDAP runtime performance tips

You can improve your ARDBC LDAP runtime performance through the use of time-based queries and caching.

Time-base queries

Use time-based queries to 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 Timestamp. Then, in the Advanced Search Bar, enter a query for records that meet your timestamp criteria. For example, use `modifyTimeStamp >= "8/9/2005 4:00:00 PM"` to consider only records modified after 4:00 PM on 8/9/05.

This query passed will be evaluated by the plug-in and it will use it to query the Directory Server so that it will return only records modified after a certain 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 if the same request was made before. If an earlier request was cached, the search results are retrieved from the cache instead of 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

You might find the following tips helpful if you have problems with your ARDBC LDAP vendor form:

- Any field (except for Display-only fields) on the vendor form must reference an LDAP attribute that exists in the specified context. For example, if you are using 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 you are troubleshooting a problem in which data is not being returned correctly, try creating a new 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 will receive ARERR (100) Entry ID list is empty, and no records will be displayed in the client.
- If more than one record has the same value, you will 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 BaseDN field, the plug-in will search from this BaseDN rather than RootDN. This will offer better performance.

AREA LDAP plug-in

The AR 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 how to configure AR System external authentication processing.



AR System7.0 supports multiple AREA LDAP configurations. The AREA LDAP adaptors configured for your AR System server are displayed in the Configuration List at the top of the AREA LDAP Configuration form. When AR System attempts to authenticate a user, it will search each LDAP configuration in the table at the top of the form.

External authentication (including chaining) only works if you set RPC (390695) *and* you select either Authenticate Unregistered Users or Cross Ref Blank Password or both in the External Authentication tab of the Server Information window.

Configuring the AREA LDAP plug-in

If you selected the AREA plug-in option during installation of the AR System server, you can configure the AREA LDAP plug-in using the AREA LDAP Configuration form in BMC Remedy User. If you did not choose the plug-in option during the original installation, you can run the AR System server installer again and select the ARDBC and AREA LDAP plug-in options.

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 these settings into the AREA LDAP Configuration form.

Note: The settings you specify in the AREA LDAP Configuration form are saved in the ar.cfg or ar.conf file.

► **To configure settings for the AREA LDAP plug-in**

- 1 Open BMC Remedy User, and log in to the AR System server as a user in the Administrator group.
- 2 Choose File > Open > Object List to display a list of forms, guides, applications, and entry points.
- 3 Select the AREA LDAP Configuration application, and click Open.
- 4 Log in to the AREA LDAP Configuration application.

The AREA LDAP Configuration form appears.

Figure 8-2: AREA LDAP configuration form

The screenshot shows the 'AREA LDAP Configuration (New)' form. At the top, there's a header bar with the title and the BMC Software logo. Below the header is a section titled 'Configuration List' containing a table with columns for Host Name, User Base, and Configuration Order. Underneath this table are several buttons: 'Clear Fields', 'Save Current Configuration', 'Delete Configuration', 'Decrease Order', and 'Increase Order'. The main body of the form is divided into two main sections: 'Configuration Detail' and 'Defaults and Mapping Attributes to User Information'. The 'Configuration Detail' section contains two groups of fields: 'Directory Service Information' and 'User and Group Information'. The 'Directory Service Information' group includes fields for Host Name*, Port Number, Bind User, Bind Password, Use Secure Socket Layer, Certificate Database*, Failover Timeout, and Chase Referral. The 'User and Group Information' group includes fields for User Base*, User Search Filter*, Group Membership, Group Base, Group Search Filter, and Default Group(s). The 'Defaults and Mapping Attributes to User Information' section contains three columns: 'User Information', 'LDAP Attribute Name', and 'Default Value If Not Found In LDAP'. This section lists various license types (License Mask, Write License, Full Text Search License, Reserved License, Application License) and their corresponding LDAP attribute names and default values.

This form provides four categories of information:

- **Configuration List** (page 155), which contains a list of the AREA LDAP configurations you have available.
- **Directory Service information** (page 156), which specifies host name, server, port, and connection information for the server you are using as the directory service.
- **User and Group Information** (page 157), which specifies search criteria for individual users and groups in the directory service. You can enter specific keywords in these fields, which will be replaced at runtime by the actual values they represent.
 - \$\\USER\$—The name of the user logging in.
 - \$\\DN\$—The distinguished name of the user logging in.
 - \$\\AUTHSTRING\$—The value that the user enters in to the Authentication String field at the time they log in.
 - \$\\NETWORKADDR\$—The IP address of the AR System client that is accessing the AR System server.
- **Defaults and Mapping Attributes to User Information** (page 158), which specifies the AR System fields, their corresponding attribute names in the directory service, and default AR System values for these fields if no value is found in the directory service.

- 5 In the **Configuration List**, select an LDAP configuration you want to edit. For each available configuration, this list displays the host name, the user base, and the order in which AR System will attempt to use the host to authenticate. When you click on a configuration in the list, the rest of the form is populated with data from that configuration.

You can click the following buttons in the Configuration List area:

- **Create New Configuration**—Clears fields in form and creates a new configuration.
- **Save Current Configuration**—Saves the current configuration with the values entered in the form.
- **Delete Configuration**—Deletes the configuration selected in the Configuration List.
- **Decrease Order**—Move the selected configuration down in the order of authentication attempt.

- **Increase Order**—Move the selected configuration up in the order of authentication attempt.

Note: If you make changes to the list of items in the Configuration List section, you must restart your AR System server.

- 6 Under **Directory Service Information**, fill in the fields shown in the following figure.

Figure 8-3: AREA LDAP configuration—directory service information area

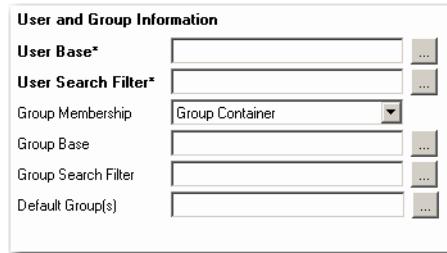
Directory Service Information

Host Name *	<input type="text"/>	[...]
Port Number	<input type="text"/>	[+]
Bind User	<input type="text"/>	[...]
Bind Password	<input type="text"/>	[...]
Use Secure Socket Layer	<input type="checkbox"/>	
Certificate Database*	<input type="text"/>	[...]
Failover Timeout	<input type="text"/>	[+]
Chase Referral	<input type="text"/>	[+]

- a **Host Name:** Enter the names of one or more servers on which the directory service is hosted. You can specify a space-separated list of hostnames up to 255 characters long. Starting with the first hostname in the list, AR System will attempt to connect to each server until it is successful.
- b **Port Number:** Enter the number of the port on which the directory service is listening.
- c **Bind User:** The distinguished name is the name for a user account that has read permissions and can search the directory service for user objects. Enter a distinguished name for this configuration.
- d **Bind Password:** Enter a password for the distinguished name specified in step c.
- e **Use Secure Sockets Layer?:** To specify an SSL connection to the directory service, select Yes. A Yes selection enables the Certificate Database field, and you can enter a certificate database in step f.
- f **Certificate Database:** If you selected Yes in the Use Secure Sockets Layer? field, enter the directory name where the certificate database file cert7.db are located.

- g Failover Timeout:** Specify the number of seconds within which the directory service must respond to the Plug-in server before an error is returned. The minimum value is 0. In this case, the connection must be made immediately. The failover timeout cannot be set higher than the value of the External-Authentication-RPC-Timeout parameter.
- 7 Under User and Group Information,** fill in the fields shown in the following figure.

Figure 8-4: AREA LDAP configuration—user and group information



- a User Base:** The base name of the search for users in the directory service (for example, `o=remedy.com`).
- b User Search Filter:** The search criteria used to locate user authentication information. Enter the keyword `\USER`—make sure that you enter a backslash after the first dollar sign—to indicate the user name for the user who is attempting to log in (for example, `uid=\USER`). At runtime, this keyword will be replaced with the user name.
- c 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 Groups fields are disabled.)
- d Group Base:** The 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 will perform a subtree search within the group you specify.
- e Group Search Filter:** The search criteria used to locate the groups to which this user belongs. Enter the keyword `\DN`—make sure that you enter a backslash after the first dollar sign—for the distinguished name of the user (for example, `uniqueMember=\DN`). At runtime, this keyword will be replaced with the distinguished name.



- 8** Under **Defaults and Mapping Attributes to User Information**, enter names for the following attributes in the LDAP Attribute Name column. In the Default AR Value if Not Found in LDAP column, select or enter a default value that will be used if a value is not found in the directory service.

Figure 8-5: AREA LDAP configuration—defaults and mapping attributes

Defaults and Mapping Attributes to User Information		
User Information	LDAP Attribute Name	Default Value If Not Found In LDAP
License Mask	<input type="text"/>	<input type="text"/>
Write License	<input type="text"/>	<input type="text"/>
Full Text Search License	<input type="text"/>	<input type="text"/>
Reserved License	<input type="text"/>	<input type="text"/>
Application License	<input type="text"/>	<input type="text"/>
Email Address	<input type="text"/>	<input type="text"/>
Default Notification Mechanism	<input type="text"/>	<input type="text"/>

- a** **License Mask:** Enter a number for the license mask. The license mask specifies whether the AREA plug-in will override existing information from the User form for write and reserved licenses, and also specifies which license types will be overridden by the value returned by the AREA plug-in. The value for the license mask is represented by an integer as outlined in the first column of the following table. An X in the column for the license type means that the value returned from the AREA plug-in will override the license that is present in the User form for the specified user.

**License Reserved Write Application
mask**

0			
1		X	
2			
3		X	
4	X		
5	X	X	
6	X		
7	X	X	
8			X

License mask	Reserved	Write	Application
9		x	x
10			x
11		x	x
12	x		x
13	x	x	x
14	x		x
15	x	x	x

- b Write License:** Select the type of license—Read, Floating, or Restricted Read.
 - c Full Text Search License:** The license type to be selected for an FTS license.
 - d Reserved License:** The license type to be selected for a reserved license.
 - e Application License:** Enter the name of the application that is licensed.
 - f Email Address:** Enter a default email address for notifications.
 - g Default Notification Mechanism:** Select the notification method used in your environment—none, alert, email, or default.
- 9** Click Save Current Configuration.

The system updates the ar.cfg or ar.conf files with the parameters you have specified in this form.

Configuring your AR system server to use the AREA LDAP plug-in

Make sure that your AR System server is properly configured to work with the AREA plug-in. Use the External Authentication tab in the Server Information window in BMC Remedy Administrator to specify AR System server-specific AREA plug-in configuration settings. See the *Configuring* guide for more information.

Configuring AREA LDAP group search

In earlier versions of AR System, 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. This means administrators had to create an AR System group for each LDAP group. Also, AR System searched for groups at only one level within the base group defined. AR System 7.0 includes enhancements that allow you to ignore excess LDAP groups, and map LDAP groups to AR System groups.

Mapping LDAP groups to AR System groups

In previous releases of AR System, the names of LDAP groups had to match the names of AR System groups for a user to be authenticated. Typically, this meant that the AR System administrator had to create an AR System group for each LDAP group. Now, you can *map* LDAP groups to AR System groups. You use the Group Mapping table on the External Authentication tab in the Server Information window to map LDAP groups to AR System groups.

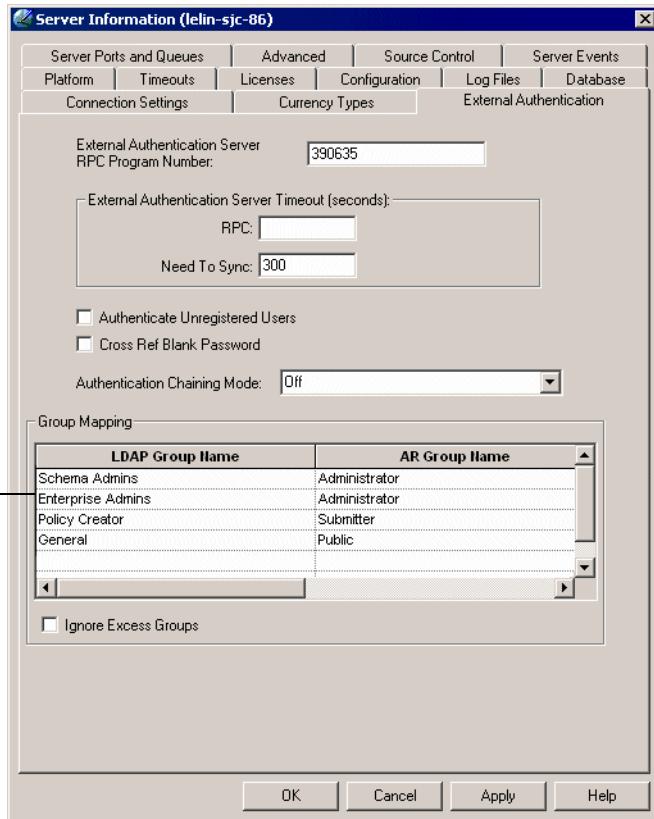
Note: Ignoring excess LDAP groups and mapping LDAP groups to AR System groups (see “Ignoring excess LDAP groups” on page 161) can be used together for maximum benefit.

► To map LDAP groups to AR System groups

- 1 In BMC Remedy Administrator, open the Server Information window and click the External Authentication tab.
- 2 Click in the Group Mapping table to add a row and type the names of the LDAP and AR System groups you want to map. You can 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, however, AR System will use only the first mapping.

Figure 8-6: LDAP group mapping table on external authentication tab



- 3 Click Apply and OK.

Ignoring excess LDAP groups

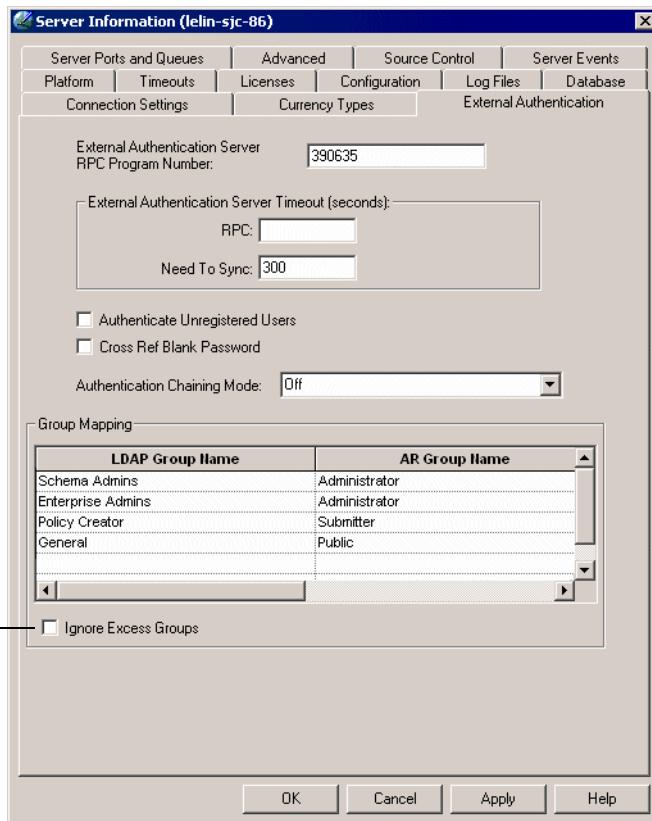
In previous releases of AR System, a user was authenticated only when *each* LDAP group to which a user belonged matched an AR System group. In AR System version 7.0, you can configure AR System to authenticate a user when *any single* LDAP group to which the user belongs matches an AR System group. You do this by specifying that AR System ignore excess LDAP groups.

Note: Ignoring excess LDAP groups and mapping LDAP groups to AR System groups can be used together for maximum benefit. See “Mapping LDAP groups to AR System groups” on page 160.

► To configure AR System to ignore excess groups

- In BMC Remedy Administrator, open the Server Information window and click the External Authentication tab.

Figure 8-7: Ignore excess groups check box on external authentication tab



- In the Group Mapping box, select the Ignore Excess Groups check box.
- Click Apply and OK.

Configuring external authentication processing

After you have configured your AR System server to use LDAP authentication, you configure AR System to use external authentication processing. See “Configuring authentication processing” on page 165.

9 AR System external authentication (AREA)

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 authentication (via the AR System User form) and external authentication, including OS-based authentication.

The following topics are provided:

- Overview of AR system external authentication (AREA) (page 164)
- Specifying AREA plug-in server configuration settings (page 165)
- Configuring authentication processing (page 165)
- Setting up the AREA hub (page 174)

Overview of AR system external authentication (AREA)

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 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 information with the user schema information. To implement external authentication, you follow these general steps:

Step 1 Create a library (.dll or .so) to handle AREA API calls. For more information, see the following sections:

- “Creating plug-ins” on page 118
- “Common plug-in API functions” on page 120
- “AREA plug-in specific API functions” on page 124.

Step 2 Link to the AREA library.

Step 3 Install the AREA library on the machine or machines that contain the AR System server.

Step 4 Using BMC Remedy Administrator, configure the AR System server to use external authentication.

About the AREA LDAP plug-in

AR System includes a sample AREA LDAP plug-in. It is installed if you selected the AREA plug-in option during installation of the AR System server. If you did not choose the plug-in option during the original installation, you can run the AR System server installer again and select the AREA LDAP plug-in option.

For more information about configuring the AREA LDAP plug-in, see “Configuring the AREA LDAP plug-in” on page 152.

Specifying AREA plug-in server configuration settings

Make sure that your AR System server is properly configured to work with the AREA plug-in. You use the External Authentication tab in the Server Information window in BMC Remedy Administrator to specify AR System server-specific AREA plug-in configuration settings. See “Configuring the AREA LDAP plug-in” on page 152 and the *Configuring* guide for more information.

Configuring authentication processing

AR System allows you to authenticate users by using internal (User form) authentication, external authentication, or a combination of the two. You configure AR System to use one type of authentication or a combination of the two; and when to use them through the use of several server options. In addition, if you specify a combination of internal and external, 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 by using the User form. In all cases, if the user and password match an existing entry in the User form, the user passes authentication. Similarly, in all cases, if the user and password do *not* match an existing entry in the User form, authentication fails.

Selecting either the Authenticate Unregistered Users option or the Cross Ref Blank Password option causes AR System to use external authentication.

Important: In order for AR System to use external authentication, you must select either the Authenticate Unregistered Users option or the Cross Ref Blank Password option and specify a value of 390365 in the External Authentication Server RPC Program Number field on the External Authentication tab in the Server Information window.

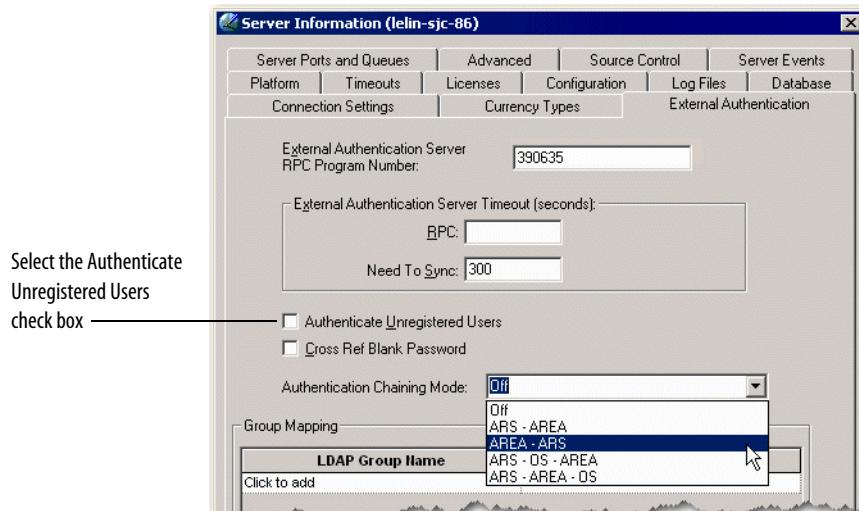
Authenticating unregistered users

If you choose the Authenticate Unregistered Users option, AR System attempts to find the user in the User form. If the user exists in the User form, AR System attempts authentication using the User form. If the user does not exist in the User form, AR System attempts authentication using the AREA plug-in.

► To enable the authenticating unregistered users option

- 1 Open the Server Information window and click on the External Authentication tab.
- 2 Select the Authenticate Unregistered Users check box.

Figure 9-1: Authenticate unregistered users check box—external authentication tab



- 3 Click Apply and then click OK.

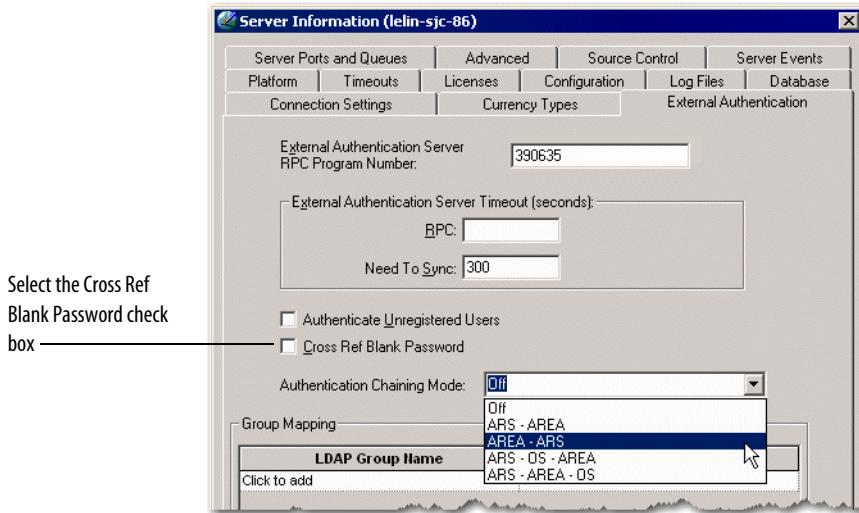
Cross referencing blank passwords

If you choose to use the Cross Reference Blank Password option, AR System attempts to authenticate using the User form if the user provided a password. If the user and password match an existing entry in the User form, the user passes authentication. If the user does not provide a password, AR System attempts to cross reference the user in an external system using the AREA plug-in.

► To enable the cross reference blank password option

- 1 Open the Server Information window and click on the External Authentication tab.
- 2 Select the Cross Ref Blank Password check box.

Figure 9-2: Cross ref blank password check box on external authentication tab



- 3 Click Apply and then click OK.

Specifying authentication chaining mode

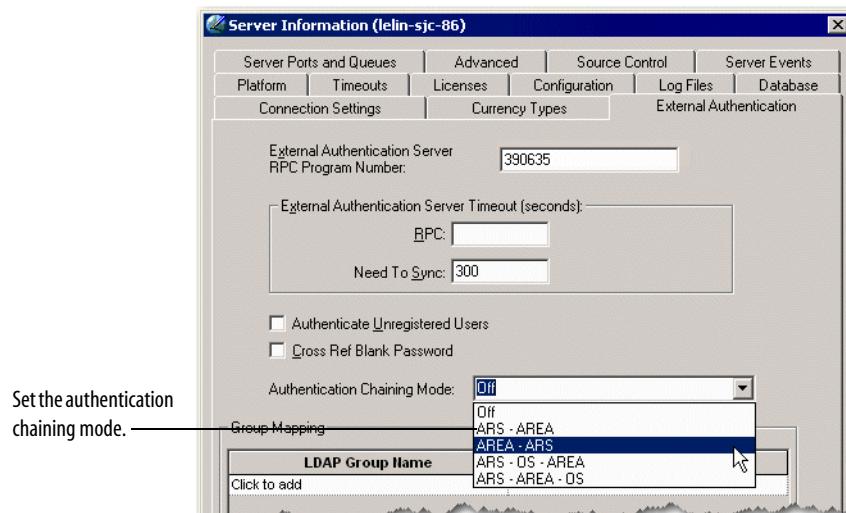
You can specify the order in which internal and external authentication methods are attempted by setting a value for the Authentication Chaining mode field.

With Authentication Chaining enabled, AR System attempts to authenticate using internal and external methods in an order you specify. All of the authentication methods in the chain are attempted, regardless of failures, until either authentication is successful or all of the authentication methods in the chain have been attempted.

► To set the authentication chaining mode

- 1 Open the Server Information window and click on the External Authentication tab.

Figure 9-3: Authentication chaining mode field on external authentication tab



External authentication (including chaining) only works if you set RPC (390695) *and* you select either Authenticate Unregistered Users, Cross Ref Blank Password, or both.

- 2 From the Authentication Chaining Mode list, select one of the following values:

Mode	Description
Off	Disables authentication chaining.
ARS - AREA	AR System attempts to authenticate the user using the User form, and then the AREA plug-in.
AREA - ARS	AR System attempts to authenticate the user using the AREA plug-in, and then the User form.

Mode	Description
ARS - OS- AREA	AR System attempts to authenticate the user 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 using the User form, then the AREA plug-in, and then Windows or UNIX authentication.

Note: AR System will behave differently depending on the authentication chaining mode you choose combined with other external authentication parameters you specify. For more information, see “Determining AREA behavior” on page 169.

- 3 Click Apply and then click OK.

Note: If you use the AREA Hub, the plug-ins installed in through the AREA Hub the authentication chaining mode treats the AREA Hub 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 174 for more information.

Determining AREA behavior

Several factors affect how AR System authenticates users, including:

- Whether Authenticate Unregistered Users is selected
- Whether Cross Reference Blank Password is selected
- The value of the External Authentication Server RPC Program Number
- Whether the user being authenticated exists in the User form and, if so, whether a password exists for the user.

The following tables describe how AR System authentication behaves for a given configuration.

With external authentication server RPC program number set to 390695

The tables in this section assume that both Authenticate Unregistered Users and Cross Reference Blank Password are selected and that the External Authentication Server RPC Program Number is set to 390695.

User with no password in User form

Authentication chaining mode	Authentication behavior
------------------------------	-------------------------

Off	<ul style="list-style-type: none">■ Authentication performed using AREA LDAP password. User information retrieved from the User form.■ Authentication process stops when it fails using AREA LDAP.
ARS - AREA	<ul style="list-style-type: none">■ Authentication performed using AREA LDAP password. User information retrieved from User form.■ Authentication process will stop when it fails using AREA LDAP.
AREA - ARS	<ul style="list-style-type: none">■ Authentication 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, then missing information in the form will be retrieved from the User_Cache.■ If AREA LDAP authentication fails, then authentication processing stops.
ARS - OS - AREA	<ul style="list-style-type: none">■ User Authentication performed using AREA LDAP. If successful, user information is retrieved from AR System.■ If AREA LDAP authentication fails, then 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	<ul style="list-style-type: none">■ User Authentication performed using AREA. If successful, user information is retrieved from AR System.■ If AREA LDAP authentication fails, then 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 does not exist in User form

Authentication chaining mode	Authentication behavior
Off	<ul style="list-style-type: none"> ■ Authentication performed using AREA LDAP. User information is retrieved from AREA LDAP.
ARS - AREA	<ul style="list-style-type: none"> ■ Authentication is <i>not</i> performed using AR System because the user does not exist in the User form. ■ Authentication performed using AREA LDAP. If successful, user information is retrieved from AREA LDAP.
AREA - ARS	<ul style="list-style-type: none"> ■ Authentication performed using AREA LDAP. If successful user information is retrieved from AREA LDAP. ■ Authentication is <i>not</i> performed using AR System because the user does not exist in the User form.
ARS - OS - AREA	<ul style="list-style-type: none"> ■ Authentication is <i>not</i> performed using AR System because the user does not exist in the User form. ■ Authentication performed using OS authentication. If successful, user information is retrieved from the OS. ■ If OS authentication fails, then user authentication is performed using AREA LDAP. If AREA LDAP authentication is successful, user information is retrieved from AREA LDAP.
ARS - AREA - OS	<ul style="list-style-type: none"> ■ Authentication is <i>not</i> performed using AR System because the user does not exist in the User form. ■ Authentication performed using AREA LDAP. If successful, user information is retrieved from AREA LDAP. ■ If AREA LDAP authentication fails, then the user will be authenticated using OS authentication. If OS authentication is successful, user information is retrieved from the OS.

User exists with password in User form

Authentication chaining mode	Authentication behavior
Off	<ul style="list-style-type: none"> ■ Authentication performed using User Form. If successful, user information is retrieved from the User form. ■ If User form authentication fails, authentication will <i>not</i> be attempted using AREA LDAP.
ARS - AREA	<ul style="list-style-type: none"> ■ Authentication performed using 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	<ul style="list-style-type: none"> ■ Authentication performed using AREA LDAP. If successful, user information is retrieved from AREA LDAP. ■ If AREA LDAP authentication fails, then authentication is attempted using AR System User form. If User form authentication is successful, user information is retrieved from the User form.
ARS - OS - AREA	<ul style="list-style-type: none"> ■ Authentication performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, then OS authentication is attempted. If successful OS authentication is successful, user information is retrieved from the OS. ■ If OS authentication fails, then AREA LDAP authentication is attempted. If AREA LDAP authentication is successful, user information is retrieved from AREA LDAP.
ARS - AREA - OS	<ul style="list-style-type: none"> ■ Authentication performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, then AREA LDAP authentication is attempted. If AREA LDAP authentication is successful, user information is retrieved from AREA LDAP. ■ If AREA LDAP authentication fails, then OS authentication is attempted. If successful OS authentication is successful, user information is retrieved from the OS.

With external authentication server RPC program number set to 0

The tables in this section assume that both Authenticate Unregistered Users and Cross Reference Blank Password are selected and that the External Authentication Server RPC Program Number is set to 0.

User with no password in User form

Authentication chaining mode	Authentication behavior
For all Authentication chaining modes	<ul style="list-style-type: none"> ■ Authentication performed using OS authentication. If successful, user information is retrieved from the User form. ■ If OS authentication fails, authentication processing stops.

User does not exist in User form

Authentication chaining mode	Authentication behavior
For all Authentication chaining modes	<ul style="list-style-type: none"> ■ Authentication 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	<ul style="list-style-type: none"> ■ Authentication 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	<ul style="list-style-type: none"> ■ Authentication 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.

Authentication chaining mode	Authentication behavior
AREA - ARS	<ul style="list-style-type: none"> ■ Authentication performed using OS authentication. If successful, user information is retrieved from the OS. ■ If OS authentication fails, then User form authentication is attempted. If AR System authentication is successful, user information is retrieved from the User form.
ARS - OS - AREA	<ul style="list-style-type: none"> ■ Authentication performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, then OS authentication is attempted. If OS authentication is successful, user information is retrieved from the OS.
ARS - AREA - OS	<ul style="list-style-type: none"> ■ Authentication performed using the AR System User form. If successful, user information is retrieved from the User form. ■ If AR System authentication fails, then OS authentication is attempted. If OS authentication is successful, user information is retrieved from the OS.

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 will be 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. For more information, see “Configuring the AREA LDAP plug-in” on page 152.

► 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_plugin.dll
```

- 3 If necessary, create entries for subsequent AREA plug-ins as follows:

```
AREA-Hub-Plugin: my_area_plugin_1.dll
```

```
AREA-Hub-Plugin: my_area_plugin_2.dll
```

```
AREA-Hub-Plugin: my_area_plugin_3.dll
```

- 4 Restart the AR System plug-in server.

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 178)
- Services provided to the data visualization modules on BMC Remedy Mid Tier (page 178)
- Services provided on clients (page 180)
- Using Java classes (page 181)
- Creating data visualization fields (page 183)

Overview



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. This field 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 will automatically download this code from the AR System server and use 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, functionReference)`. 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 within 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 quotes if it contains spaces (with the AR System standard of escaping of double quotes being two double quotes together, representing a single double quote 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\$/ \$EVENTWINSRID\$ keywords. The \$EVENTTYPE\$ and \$EVENTDATA\$ keywords will be set to the values passed in the call. The EVENTSRCWINID will be set to F<module fieldID>.
- 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 the data visualization developers. The Javadocs contain more information about these classes.

Note: When searching for a class name, remember that each class is present in the `com.remedy.arsys.plugincontainer` package. So, the `Plugin` class is named `com.remedy.arsys.plugincontainer.Plugin`.

Interfaces that data visualization developers can implement:

- **Plugin**—An interface that must be implemented by data visualization developers.
- **DefinitionFactory**—An interface that can optionally be implemented by data visualization developers to parse their definition data into Java objects. These Java objects must be imported into the **Definition** or the **CacheableDefinition** interfaces described in the list that follows.
- **Definition**—A marker interface to be implemented by the modules' definition objects.
- **CacheableDefinition**—A marker interface to use if the module needs the container to cache the definition objects. The cached objects will be returned when the module requests the **DefinitionService** to get the definition. For further information, see the cache service.

Other interfaces include:

- **ARPluginContext**—Extends the **PluginContext** class to provide AR System specific services.
- **AuthenticationException**—Extends the **Exception** class.
- **CacheableDefinition**—Implemented to use the caching services of the plug-in container.
- **DefaultDefinition**—Default implementation of the **Definition** class.
- **Definition**—Used 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 allow the module writer to localize the plug-in. For example, you can format dates, time, and numbers for locale conventions.
- **LoginContext**—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**—Used to generate URLs that provide a reference back to the data visualization.

Creating data visualization fields

Use the following process to create a data visualization field.

- Step 1** Create a module on the mid tier. (See page 183.)
- Step 2** Register the module. (See page 187.)
- Step 3** Deploy a custom data visualization module. (See page 188.)
- Step 4** Add a data visualization field to a form (See the *Form and Application Objects* guide.)

Creating data visualization modules on the mid tier

Use the following procedure to create data visualization modules.

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 Sun Java `HttpServletResponse` class in your `-classpath` parameter for `javac`. `HttpServletResponse` is in `j2ee.jar` if you are using a SunOne web server; it is in `servlet.jar` if you are using `ServletExec`. For example:

```
javac -classpath <your directory structure>/j2ee.jar:<your  
directory structure>/GraphPlugin.jar HelloWorldPlugin.java
```

- 3 Add the compiled .class file to a .jar file. For example:

```
jar -cvf DataVisHelloWorld.jar HelloWorldPlugin.class
```

If you are using a package structure, enter:

```
jar -cvf DataVisHelloWorld.jar <top level directory>
```

Remember the following guidelines if you are using package structures:

- 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. So, if you need to see files in directories above your current directory, use the .. / reference; you cannot simply use an absolute file path reference.

Following are examples that you can use to create data visualization modules.

Example: HelloWorld plug-in

This example sends back HTML that will display 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 of 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(ps.getEventDispatcherName()+" .sendEventToMidTier(\"");
writer.println("ClickEvent\", \"Title\");");
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 will be 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 will be 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, String
    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

Data visualization modules must be registered on 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 Registration form in New mode.

This form tells the mid tier server that the plug-in exists.

Figure 10-1: Data visualization module registration form

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 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 data visualization modules are registered, the version with the latest version number will be 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 data visualization module 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 Remedy User, open the Data Visualization Definition form in New mode.
This form defines the instance of your plug-in.

Figure 10-2: Data visualization definition form

The screenshot shows the 'Data Visualization Definition' form. It includes the following fields:

- Request ID:** A text input field.
- Definition Name:** A text input field containing "HelloWorld". To its right is a button with three dots.
- Sub Type:** A text input field.
- Description:** A text input field containing "hello world". To its right is a button with three dots.
- Module Name:** A text input field containing "HelloWorld". To its right is a button with three dots.
- Status:** A dropdown menu showing "Active".
- Modified Date:** A text input field.
- Definition:** A section with the sub-instruction "Either type a simple definition, or attach a more complex definition". It contains two sections: "Simple Definition" (with a text input field and a three-dot button) and "Complex Definition" (with a table).

File Name	Max Size	Attach Label
DataVisHelloWorld.jar	3 KB	Definition(Big)
- Permissions:** A section with four groups:
 - Groups & Roles:** A text input field.
 - Application:** A text input field.
 - Final Groups:** A text input field.
 - GUID:** A text input field.
 Each group has a three-dot button to its right.

- 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 (linking it to your data module on the Advanced tab of the Data Visualization field properties), as described in the *Form and Application Objects* guide.

If you want 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 Server Information window in BMC Remedy Administrator).

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 192)
- Creating vendor forms (page 193)

About vendor forms

Vendor forms allow the 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 `arplugin` 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. For more information about ARDBC plug-ins, see “ARDBC Plug-Ins (AR System database connectivity) and vendor forms” 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:

- A vendor form cannot be used in join forms.
- You can add only Required and Optional fields to the Vendor form that correspond to actual columns within the data source. In addition, you can add a Display Only field only when the column name does not correspond to column within the data source.
- Attachment fields are not supported on vendor forms.
- 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 *Configuring* guide.

Note: Creating a vendor form for an ARDBC LDAP plug-in is a special case.

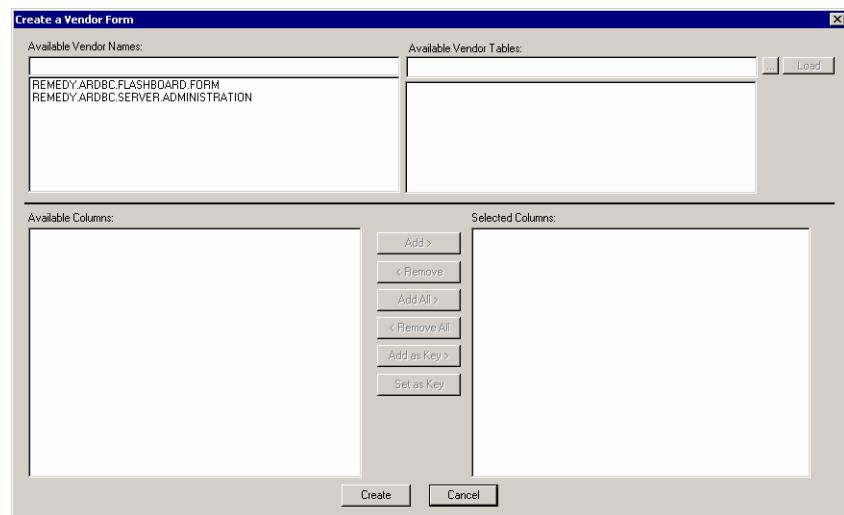
For more information, see “Creating a vendor form to represent a collection of LDAP objects” on page 146.

► To create a vendor form using an ARDBC plug-in

- 1 From the New Server Object dialog box in BMC Remedy Administrator, select Vendor Form.

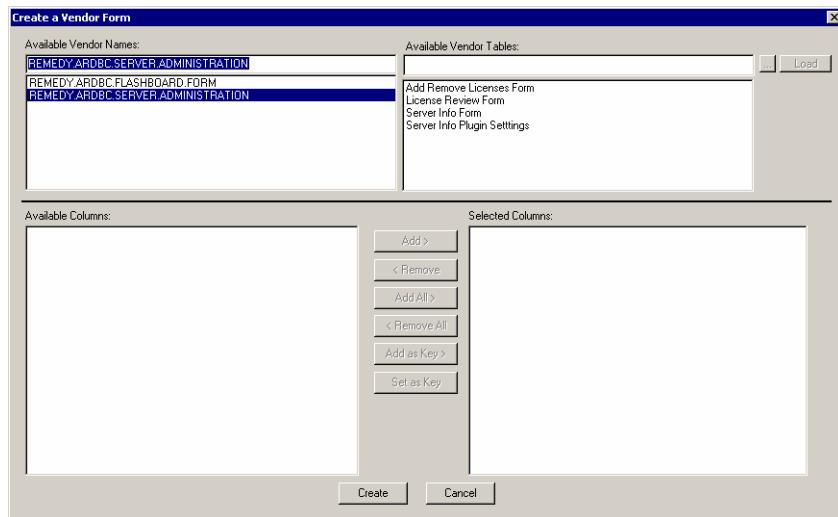
The Create a Vendor Form dialog box appears.

Figure 11-1: Create a Vendor Form dialog box



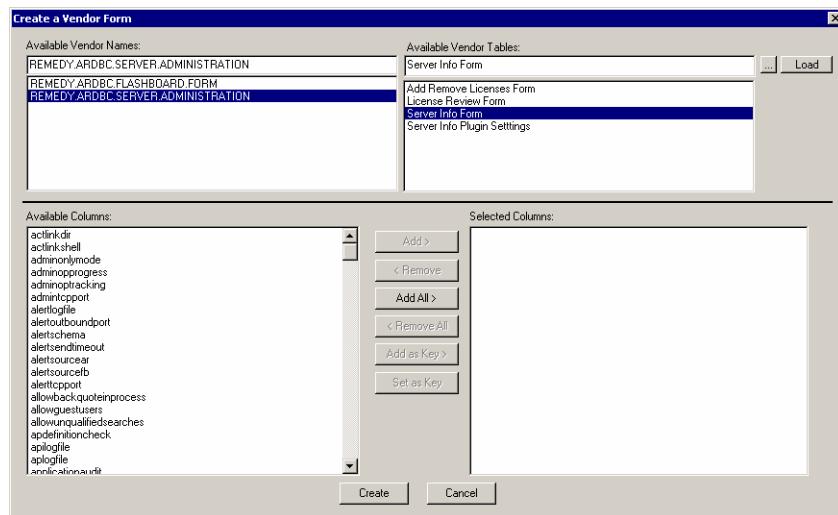
- 2 Choose the ARDBC plug-in that you want to use in the list of Available Vendor Names.

The names of the tables available from the vendor name you selected appear in the Available Vendor Tables field.

Figure 11-2: Available Vendor Tables field populated

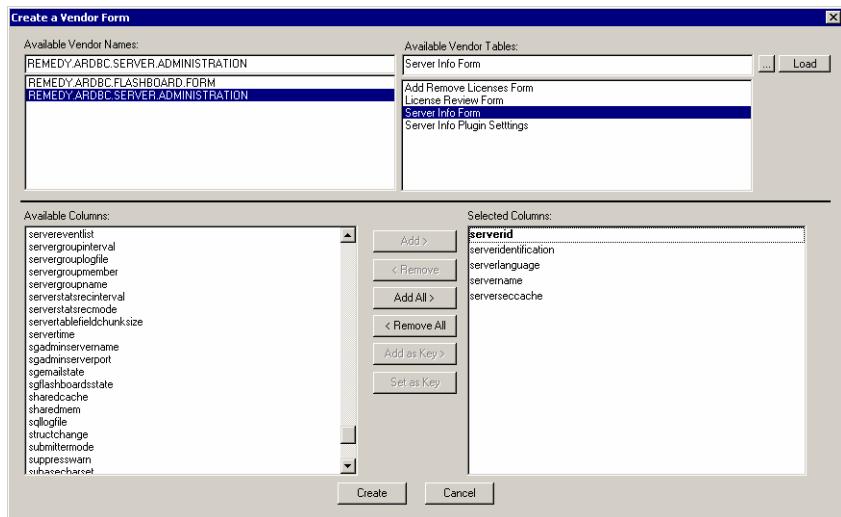
- 3 To obtain the list of attributes for the table and refresh the list of Available Columns from a randomly typed URL, click the Load button.
- 4 Choose the remote data table that you need to access in the list of Available Vendor Tables.

The fields in the chosen table appear in the Available Columns list.

Figure 11-3: Available Columns Field populated

- 5 Select the columns that you want to access in the AR System, and click Add. The columns you select appear in the Selected Columns field.

Figure 11-4: Selected Columns Field populated



- 6 Select a key column to uniquely identify the entries in your vendor form. The values in the key column are mapped to the Request ID field on the Vendor Form. Perform one of the following actions:
- Select a column already in the Selected Columns list and click Set as Key.
 - Select a column from the Available Columns list and click Add as Key.

Note: In previous releases, AR System required that the column you select as unique identifier contain no more than 15 characters. With release 7.0, you can select a key containing more than 15 characters. However, to display a key greater than 15 characters in length, you must also add another field to the form that is associated with the same attribute and set the results list to use this field rather than the Request ID. This is because even though the request ID will function with data greater than 15 characters on vendor forms, it is a core field whose size cannot be altered. For more information, see “Identifying objects uniquely” on page 148.

- 7 When you finish, click Create, and the Vendor Form and table will be created and will appear in BMC Remedy Administrator for you to edit.

Chapter 12 View forms

View forms are AR System objects that enable the AR System server to access external relational databases through AR System forms. View forms allow AR System to point to, and access, data in an existing database table created outside AR System. The table can be located on the same database instance, or in any other databases accessible from the current AR System database.

The following topics are provided:

- Overview (page 198)
- Setting up a remote database for view forms (page 202)
- Field properties for fields on view forms (page 203)
- Modifying fields on view forms (page 204)
- Issues and considerations (page 204)

Overview

When creating view forms, remember the following guidelines and limitations:

- The database table must reside on, or be accessible to, the database that AR System is using.

Note: For DB2 databases, you can only connect to database tables in the AR System database.

- 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 will act as the Request ID. If the administrator chooses a column that is non-unique or allows nulls, data corruption will probably occur. The Request ID field must be an integer field or a character field that is no less than 6 and no greater than 15 characters. Otherwise, the Key field list will be empty, and you will not be able to create the view form.
- A view form can be manipulated as a regular form type with the following exceptions:
 - You can only add Required and Optional fields to the View form that correspond to actual columns within the external table. In addition, you can add a Display Only field only when the column name does not correspond to column within the external table.
 - A view form cannot be used in join forms.
 - After you have attached 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.
 - FTS operations are not available.

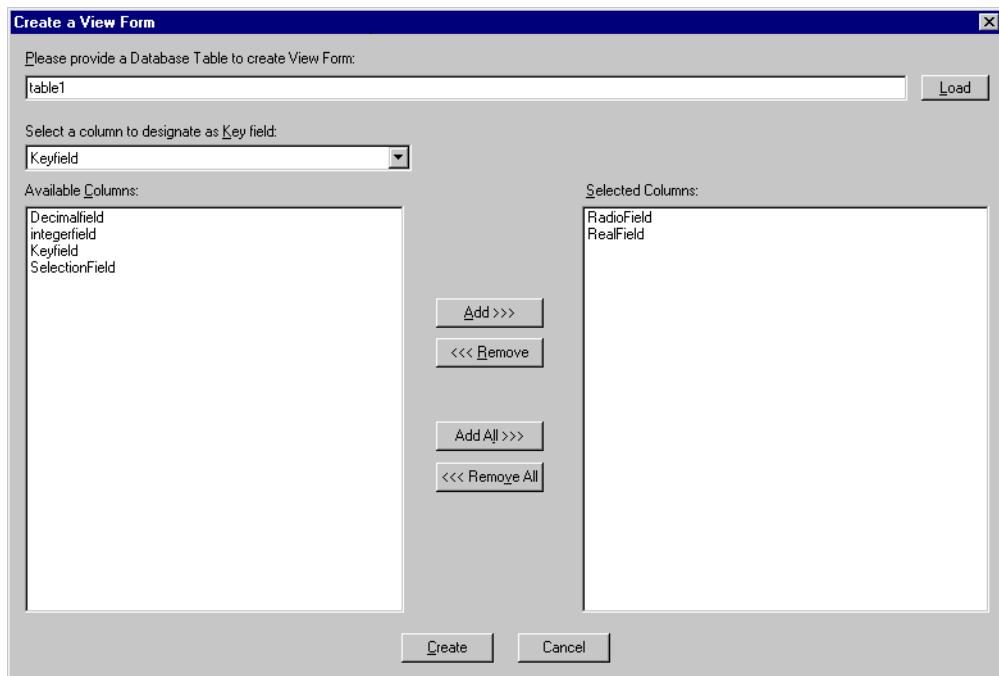
- Only the following data types are supported in view forms:
 - **DB2**—varchar, varchar2, char, clob, integer, smallint, real, double
 - **SQL Server**—char, varchar, tinyint, smallint, int, text, real, float, decimal
 - **Oracle**—varchar, varchar2, char, clob, number, float
 - **Informix**—char, varchar, int8, smallint, integer, text, smallfloat, float, decimal
 - **Sybase**—char, varchar, tinyint, smallint, int, text, real, float, decimal
- You cannot use a view form to access LOBs on a remote database.
- Long columns (that is, text or clob) must allow null values.

The steps that follow explain how to connect to a database table using a view form.

► To create a view form

- 1 If your database is remote, set it up as described in “Setting up a remote database for view forms” on page 202.
- 2 In BMC Remedy Administrator, select a server to administer.
- 3 Choose File > New Server Object.
- 4 Select View Form from the list to open the Create a View Form dialog box.

Figure 12-1: Create a view form dialog box



- 5 Enter the name of an existing database table or the path if the table is on a different database.

The formats for table names are as follows:

- **DB2—TABLENAME**

Table name for the AR System database. For DB2 databases, you can only connect to tables in the AR System database.

By default, use all capital letters when entering the table name, because DB2 defaults to all capital letters for the data within its system tables.

Note: For an AR System server talking with a *local* AR System database running on DB2, in the ar.conf (ar.cfg) file, you must set the Db-user option to the name of the user who owns the database. For example, if the owner name is db2admin, make sure that the following line is in the ar.conf (ar.cfg) file: Db-user: db2admin

For more information about the ar.conf (ar.cfg) file, see the *Configuring* guide.

■ **Informix**—tablename or databasename:table

Table name for a remote Informix database. By default, use all lowercase letters when entering the owner and table name, because Informix defaults to all lowercase letters for the data within its system tables.

■ **Oracle**—OWNER.TABLENAME or TABLENAME

Oracle defaults to all capital letters for the data within its system tables. If the table was created with a case-sensitive name, make sure that the capitalization for the name is entered correctly.

■ **SQL Server**—TABLENAME or DATABASENAME.OWNER.TABLENAME

Make sure you specify dbo if the current user is the owner of the table.

■ **Sybase**—TABLENAME or DATABASENAME.OWNER.TABLENAME

Make sure you specify dbo if the current user is the owner of the table.

6 Click the Load button.

The Available Columns list box is populated with the database column names from the external table.

7 Select a column to designate as the key field.

You must choose a character column whose name is from 6 to 15 characters or an integer field.

WARNING: The column you choose for the key field must be unique and non-null; otherwise, data corruption will probably occur.

8 Select the columns you want to appear on the AR System form in the Available Columns list box, and click Add to move them to the Selected Columns list box.

If you want to use AR System data types (date/time, date, selection, timestamps), create the fields in the form itself. Do not select them from the list as described in this step. Put the database column name in the Name field in the View Information section on Database tab.

9 Click the Create button.

Note: Creating multiple fields from a single column will have adverse side effects when accessed.

Setting up a remote database for view forms

If your database is remote, you must set up your database as follows to make sure that view forms are created properly.

- **DB2**—Remote view forms are not supported for DB2.
- **Informix**—Make sure that the table name is in the following format:
DATABASENAME@SERVERNAME : TABLENAME
- **Oracle**—Set up a link between the AR System database and the Oracle database. The AR System server must query the metadata to obtain specific information about the table, and this resides in one of the system tables. Therefore, the link must be a name to the entire server, not to the specific table.

The format for the table name is as follows:

OWNERNAME . TABLENAME@LINK

To enable the support of multiple remote Oracle databases with different character sets, you must add the parameter `Oracle-Dblink-Character-Set` to your `ar.cfg` (`ar.conf`) file. See the *Configuring* guide for more information.

- **SQL Server**—Complete the following tasks:
 - Create a link to the remote database and either give ARAdmin an account on the remote database or use the proper login credentials.
 - Turn on Distributed Transaction Coordinator for both the local and remote database.
 - Set the following server configuration setting in the `ar.conf` (`ar.cfg`) file:
`SQL-Server-Set-ANSI-Defaults: T`

This allows the DB-Library connection that the AR System uses to use ANSI-NULLs as well as ANSI warnings. This should have no impact on the performance of the database. For more information about the `ar.conf` (`ar.cfg`) file, see the *Configuring* 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. Consult the Sybase documentation for details on how to create a proxy database. After the proxy database is created, it can be accessed like a local database in the form of DATABASENAME.OWNER.TABLENAME.

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 Database tab in the Field Properties dialog box has an additional section with the following fields:
 - **Table**—Indicates the link to the external database table.
 - **Name**—Displays the column name on which the field was created. For view and vendor form types, only the Name field can be modified, and it must be 254 characters or less.
- If the column name does not correspond to a column in the data source, the field must be display only.

For example, you add a character field to a view form. The Name field on the Database tab shows the column name as `Character_Field`, which does not exist in the data source. To save the form, you must change the Name to match a column in the data source, or set the Entry Mode property to Display Only.

- If the column name represents a column in the data source, the field cannot be display only.
- After initial creation, you cannot change the type of a text field, or change the length of any field.

Modifying fields on view forms

- To delete a field from the view form, click on a field and choose Edit > Delete. Deleted fields are returned to the Available Columns list box. This deletes only the AR System field. It does not remove the column from the database table.
- To add a field to the view form, choose Form > Create a New > Field From *<external_name>* from the menu. Select the field you want to add from the Add Field dialog box, and click OK.

Issues and considerations

Keep the following issues in mind when working with view forms:

- A view form cannot be used in join forms.
- FTS (Full Text Search) operations are not available on view form fields.
- You can only add Required and Optional fields to the view form that correspond to actual columns within the data source. In addition, you can only add a Display Only field when the column name does not correspond to a column within the data source.
- Attachment and status history fields are not supported on view forms.
- 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

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 206)
- Pushing data from AR System with SQL (page 206)
- Pulling data into AR System with SQL (page 207)
- Issues and considerations (page 208)

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, since there is no way to ensure data integrity. In addition, external applications reading AR System data from the database directly will not be subject to AR System permissions, nor will they trigger any AR System workflow. If this is not acceptable, then data should be read through the AR System API.

For detailed information about the AR System database, see the *Database Reference* guide.

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 will perform no pre- or post-processing on the SQL command or the results. It is up to the administrator to make sure that the command is correct. When the action is triggered, the AR System server will pass 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. If you need to run a set of commands, create separate actions, or create a stored procedure and run that. Note that stored procedures do not return values.
- Turn on AR System server SQL logging to debug the SQL syntax if it returns unexpected values or results. A good debugging strategy is to start an SQL interpreter (for example, isql for Sybase, SQL*Plus for Oracle, Command Center for DB2, or Query Analyzer for 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. If you need to access databases that are external to this DB server, then 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:

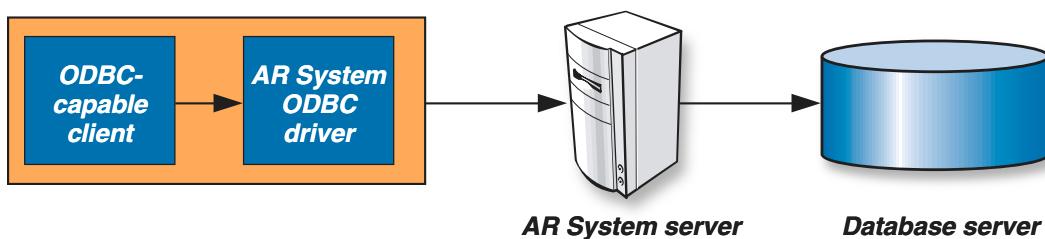
- Overview (page 210)
- Creating multiple data sources (page 211)
- Compatibility with ODBC clients (page 214)
- Using Crystal Reports with AR System (page 214)
- Using Microsoft Access with AR System (page 221)
- Using Microsoft Excel with AR System (page 222)
- Issues and considerations (page 223)

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

When you install BMC Remedy User, the mid tier, or the ARWebReportViewer, the AR System ODBC driver (`arodbc70.dll`) is installed. For example, to run Crystal Reports with any clients (a browser or BMC Remedy User), you must have the AR System ODBC driver on the machine you are using.

The AR System ODBC is installed with BMC Remedy User unless you deselect that option during installation. 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, you can create a data source called AR System Report User for access to 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 AR System 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.

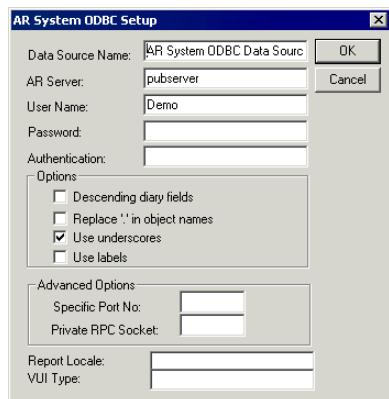
This utility is found in different locations based on the version of Windows you are using.

- 2 On the System DSN tab, select AR System ODBC Data Source, then 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

- 4 Type a unique name for the Data Source in the Data Source Name field.
- 5 Type the AR System server name for the server to be accessed with this data source in the AR Server field.
- 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 (For mid tier only) Select the Use Underscores check box if there are field names or form names in your reports that contain special characters, such as dot (.), hyphen (-), plus sign (+), and semicolon (;). When you select the Use Underscores option, underscores are used to replace special characters.

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 are using Microsoft Access, spaces and hyphens are not allowed in object names.

-
- 10 Select the Use Labels check box to use field labels based on the locale you specify in the Report Locale field. For more information, see “Using field labels or database names in Crystal Reports” on page 218.

Note: It is recommended that you *unselect* the Verify On First Refresh report option in Crystal Reports. Then, you do not need to match the Use Labels option for the report to run correctly.

If the Verify On First Refresh option is selected, you must match the Use Labels option when you create the report and at runtime. For example, if you select the Use Labels option when you create the report, you must also select it when you run the report. Conversely, if you unselect the Use Labels option when you create the report, you must also unselect it when you run the report

- 11 (For mid tier only) In the Report Locale field, enter the locale for the language in which you want to see the report.

Note: If you have installed two localized views (for example, German and French), and you are using the German localized view and the report locale setting is set to the French locale, the data returned will be in French, though the static report text will be 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 (For 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.

Note: 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 shown 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 details, see the *Installing* 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, the error ODBC error: Unexpected extra token: <xxx> (<xxx> is the name of the form) might display.

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.

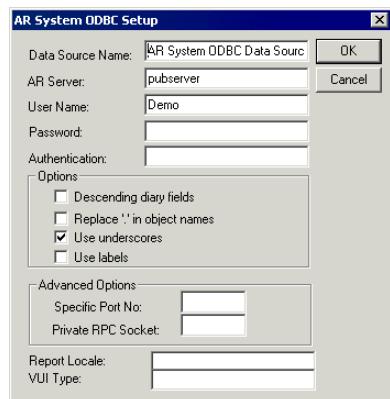
► To create a report by logging in to AR System from Crystal Reports

- 1 Open Crystal Reports and create a new report.
- 2 In the Crystal Reports Gallery, choose a wizard such as Standard.
- 3 In the Available Data Sources, choose ODBC (RDO).
- 4 When the ODBC (RDO) dialog box appears, select the Data Source you want 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



- 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.

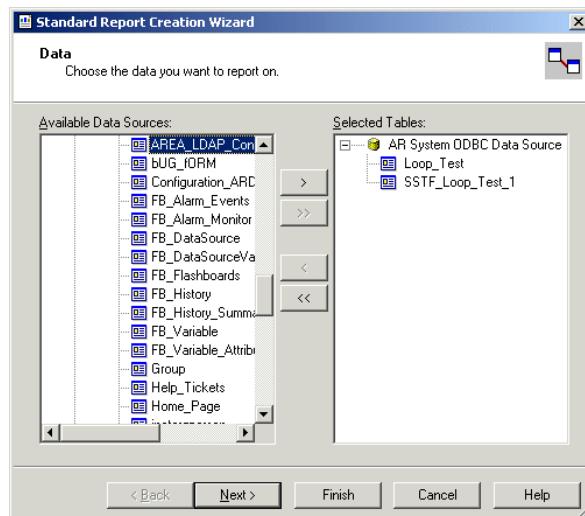
For more information, see “Using field labels or database names in Crystal Reports” on page 218.

Note: Field labels are based on the locale specified in the Report Locale field.

- 9 Click OK to log in.

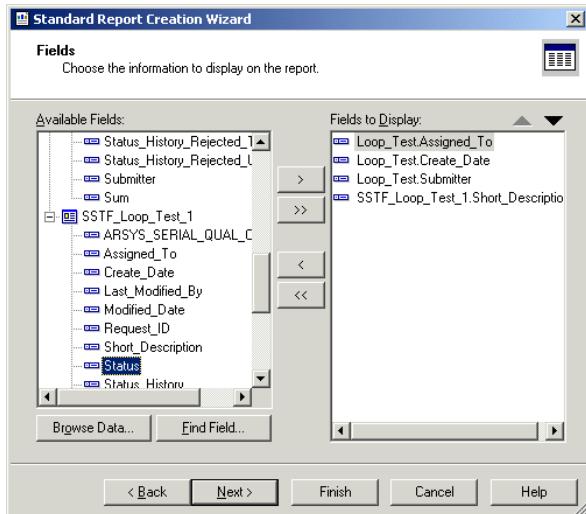
The AR System forms appear in the Standard Report Creation Wizard as data sources.

Figure 14-4: Selecting data sources in the Standard Report Creation wizard



- 10 Select the form that you want to include in your report, then click Next.
- 11 Click Next.

The wizard lists all fields in the underlying form.

Figure 14-5: Selecting fields in wizard

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. For more information, see “Using field labels or database names in Crystal Reports” on page 218.

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 Administrator, 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 Choose which fields to display in the report, and then click Next.

The wizard now lets you specify how you want to group the information to display on your report. This step is optional.

- 13** Group the information, then 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 then click Next.

The wizard now lets you specify a report template. This step is optional. Clicking one of the available templates lets you “preview” how your report will appear.

- 15** Select a template, and then click Finish.

See your Crystal Reports documentation for more details about designing reports.

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 you want to report from a list displayed by the 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 you want AR System fields to be represented in the 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 211.
- Create a report in Crystal Reports. See “To create a report by logging in to AR System from Crystal Reports” on page 215.

If you specify using field labels, the list of fields in the 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 the Crystal Report Designer displays database names for the fields.

Important: It is recommended that you use the same setting both for setting up your AR System ODBC and 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 Administrator, and 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 choose to use this type of verification, Crystal Reports will only report on field names for which there are matches between the report definition and the data source as it is configured at run time.

This means that if you change your AR ODBC configuration (that is, toggle the Use Labels check box) between the time you designed the report and the time you run the report *and* you choose to verify report fields against the AR ODBC data source, your report might not contain the data you expect it to contain and you might encounter 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.

Note: To identify a field's database name, open the form in BMC Remedy Administrator, and open the Field Properties dialog box for the field. The Name field of the Database tab displays the field's database name.

► To select the field

- 1 Choose the Fields tab on the Create Report Expert dialog box.
- 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.

See your Crystal Reports documentation for more details about designing reports.

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; however, you can 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 having 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 if they are used in your report. By selecting the *Convert to Date* option in the Reporting tab of the File Options dialog box, you are specifying that Date/Time strings from AR System are to be 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 of *is equal* (in the Select tab of the Create Report Expert dialog box in Crystal Reports). The AR System Date/Time field only works with the Convert to String or Keep Date-Time Type options.

- **List Sorting**—Consider that selection fields from AR System are treated as character types. List sorting in Crystal Reports is based on display value (New, Assigned, Closed), rather than the numeric values (0, 1, 2) associated with an enumerated field. This occurs because when the AR System ODBC driver is used, selection fields with AR_DATA_TYPE_ENUM data types are mapped to SQL_CHAR data types. 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, which typically is the Gregorian calendar starting from October 15, 1582. If the date field contains a BC date, this will not be supported by Crystal Reports.

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 212.

- 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 are working 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 will 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 timestamp) is removed with the control characters.

Note: Microsoft Excel has a date system that begins January 1st 1900. If your date field contains a BC date, it will not be supported by Microsoft Excel.

Issues and considerations

Keep the following issues in mind when working with AR System ODBC:

- The AR System ODBC driver is read-only. ODBC clients that create, modify, or delete entries or tables will 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 are unable to create an AR System join form for the data that 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. (Note that BMC Remedy User does not support multiple AR System data sources. Therefore, if you created a report using a third-party ODBC client and joined two tables directly in an SQL statement, the report would not run from AR System workflow or from the BMC Remedy User Reporting window.)

- Hidden 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.
- 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 sure that form and field names adhere to SQL standards by removing spaces and other non-standard characters.

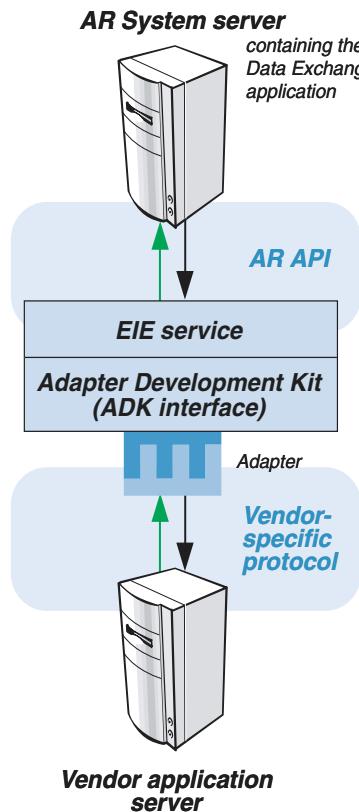
To determine which fields are in conflict, you can enable ODBC Tracing and look through the logs, or you can navigate through the Fields list in BMC Remedy Administrator to see if there are fields that meet the preceding conditions.

Chapter 15 Enterprise integration engine (EIE)

The Enterprise Integration Engine (EIE) provides the hooks to allow data to be passed 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.

EIE consists of the Data Exchange application and the EIE 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 EIE 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 15-1 illustrates this process.

Figure 15-1: Enterprise Integration Engine



The adapters communicate with EIE using the Adapter Development Kit (ADK) interface. The ADK contains the following items:

- A class library defining the interface between the EIE service and the adapter
- A template be used to create your custom adapter
- An adapter test utility to test the functionality that must be implemented
- An InstallShield script to help build an installer for your custom adapter
- A sample implementation of an adapter for a flat-file data source

The *Remedy Enterprise Integration Engine Administrator's Guide* explains how to install, configure, and maintain EIE.

The *Remedy Enterprise Integration Engine Adapter Development Kit Developer's Guide* contains detailed instructions for creating adapters.

Chapter 16 Command-line interface (CLI)

The command-line interface for the AR System clients deliver limited functionality without a graphical interface. The command-line interface is useful for writing scripts to perform repetitive operations.

The AR System command-line interfaces are used to integrate with AR System clients and, indirectly, with servers.

The following topics are provided:

- Overview (page 228)
- Using the BMC Remedy Administrator CLI (page 229)
- Using the BMC Remedy User CLI (page 236)
- Using the runmacro CLI (page 237)
- Using the BMC Remedy Import CLI (page 240)

Overview

When a computer program is started, the name of the program executable is passed to the operating system, and the application is executed. Typically, a user interface is then displayed, and the user proceeds to interact with the application. Many applications also allow parameters to be added to the command that starts the application so that the application can automatically perform some functions before the user interface is displayed. An application that supports adding parameters to the start-up command is defined as having a command-line interface (CLI).

Many applications provide the option to launch another application. If the second application has a full CLI, the first application can pass parameters to the second as a part of the launch process. This provides an easy method for applications integration and data sharing.

Several AR System clients support the CLI, which enables them to be launched and controlled by third-party applications or processes. These clients include:

- BMC Remedy Administrator (`aradmin`)
- BMC Remedy User (`aruser` and `runmacro`)
- BMC Remedy Import (`arimportcmd`)

Using the BMC Remedy Administrator CLI

This section discusses using the BMC Remedy Administrator command line interface (CLI). It includes guidelines for using the CLI and options.

Guidelines for using BMC Remedy Administrator CLI

Use the following guidelines to run commands from the BMC Remedy Administrator CLI. Commands and options are listed in Table 16-1 on page 230.

- On a machine with BMC Remedy Administrator installed locally, set the current directory to the directory that contains the BMC Remedy Administrator executable `aradmin.exe`. The default directory is `C:\Program Files\AR System\`. File arguments without a directory path are created in the current directory.
- To enter AR System CLI commands from any directory, add the BMC Remedy Administrator directory to your system path environment variable. See your Windows documentation for instructions.
- Every command executed opens a separate AR System session, executes the command, and logs out.
- BMC Remedy Administrator parses commands in the precedence listed in Table 16-1 on page 230. Commands are interpreted in a predictable order and might not be executed in the order you type 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.
- Except for the Synchronize Search Database option (`-s`), each command has required arguments. Arguments must follow the associated option.

BMC Remedy Administrator commands and options

Use the following format when running aradmin.exe. (The items between square brackets are optional.)

```
aradmin -u <user_name> [-p <password>] -x <server_name>
[-w <external_authentication_string>] [-portnum <TCP_port_number>]
[-e <file_name>] [-i <file_name>]
[-convert [-<option> -f <form_name>]]
[-c <file_name>] [-s][-o [<log_file_name>]]
```

Table 16-1: aradmin command options

Option	Description and available nested options
-u	User name—Required login parameter that identifies the user account.
-p	Password—Optional login parameter that identifies the user account. Omit the option if the user account has no password.
-x	Server name—Required login parameter that specifies the server to log in to.
-w	Authenticator—Specifies the name of an external authentication string or Windows domain. This is related to the Login window's Authentication field, which is discussed in the <i>Configuring</i> guide.
-portnum	TCP port number used to log in when the portmapper is turned off.

Table 16-1: aradmin command options

Option	Description and available nested options
-e	<p>Definition file—Specifies the file to which to export object definitions from the source server (identified with the -x login parameter).</p> <p>Available options:</p> <ul style="list-style-type: none"> ■ -a <active_link_name> for an active link ■ -A for all active links ■ -b <DSO_pool_name> for a DSO pool ■ -B for all DSO pools ■ -d <DSO_mapping_name> for a single DSO mapping ■ -D for all DSO mappings ■ -f <form_name> for a form ■ -F for all forms ■ -g <active_link_guide_name> for an active link guide ■ -G for all active link guides ■ -h <filter_guide_name> for a filter guide ■ -H for all filter guides ■ -j <type_ID> <object_name> for extension objects ■ -J for all extension objects ■ -k <packing_list_name> for a packing list object ■ -K for packing list objects ■ -l <packing_list.xml> for an XML packing list (requires argument for XML file name) ■ -m <menu_name> for a menu ■ -M for menus ■ -n <application_name> for an application ■ -N for all applications ■ -q <escalation_name> for an escalation ■ -Q for all escalations ■ -t <filter_name> for a filter ■ -T for all filters ■ -z <web_service_name> for a web service ■ -Z for all web services

Table 16-1: aradmin command options

Option	Description and available nested options
-i	<p>Source file—Specifies the file that contains the object definitions to be imported to the target server (identified with the -x login parameter). The -i option requires a <file> argument to identify the source file.</p> <p>Available options:</p> <ul style="list-style-type: none"> ■ -a <active_link_name> for an active link ■ -A for all active links ■ -b <DSO_pool_name> for a DSO pool ■ -B for all DSO pools in a server ■ -d <DSO_mapping_name> for a DSO mapping ■ -D for all DSO mappings ■ -f <form_name> for a form ■ -F for all forms ■ -g <active_link_guide_name> for an active link guide ■ -G for all active link guides ■ -h <filter_guide_name> for a filter guide ■ -H for all filter guides ■ -i inplace to overwrite existing objects, without deleting objects first ■ -j <type_ID> <object_name> for extension objects ■ -J for all extension objects ■ -k <packing_list_name> for a packing list object ■ -K for all packing list objects ■ -l <packing_list_name.xml> for an XML packing list ■ -m <menu_name> for a menu ■ -M for menus ■ -n <application_name> for an application ■ -N for all applications ■ -q <escalation_name> for an escalation ■ -Q for all escalations ■ -t <filter_name> for a filter ■ -T for all filters ■ -z <web_service_name> for a web service ■ -Z for all web services

Table 16-1: aradmin command options

Option	Description and available nested options
-convert	Use to convert native views to web views with unique names. Available options are: <ul style="list-style-type: none">■ -f <form_name> for a form■ -F for all forms■ -v <view_name> for a single view■ -V for all views■ -option <XML_file_name.xml>
-c	Use to obtain configuration details about a server. <i>All</i> configuration details for the specified server are sent to a file, including information not normally displayed by BMC Remedy Administrator. The -c command requires a <file> argument to identify the target file.
-s	Use to synchronize the search database.
-o	Use to redirect errors normally displayed within the command line interface so that they appear in a file. The -o option requires a <log_file_name> argument.
-reindex	Initiates a full text search re-index. This works the same as the option under the Full Text Search tab in the Server Information window of BMC Remedy Administrator.

Extension objects

The syntax for importing or exporting an extension object is as follows:

```
aradmin.exe /u <user> [/p <password>] /x <server> /e <exportfile>.def  
(or /i <importfile>.def) /j <type_ID> "obj1" "obj2" "obj3"
```

The type IDs for Flashboards are:

- Flashboards: 32794
- Data Sources: 32795
- Variables: 32796
- Alarms: 32797

BMC Remedy Administrator CLI examples

The following are examples of common tasks you might perform with aradmin.exe.

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:

```
aradmin -u <user_name> [-p <password>] -x <server_name>
-e <target_file_name> -F
```

To export objects from a single form, use the following command format:

```
aradmin -u <user_name> [-p <password>] -x <server_name>
-e <target_file_name> -f <form_name>
```

To parse an XML packing list, and export all objects defined in that packing list, use the following command format:

```
aradmin -u <user_name> [-p <password>] -x <server_name>
-e <target_filename> -l <packing_list.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:

```
aradmin -u <user_name> [-p <password>] -x <server_name>
-i <source_file> -F
```

To import specific objects from a source file, use the following command format:

```
aradmin -u <user_name> [-p <password>] -x <server_name>
-i <source_file> -f <form_name> -a <active_link_name>
```

To parse an XML packing list, and import all objects defined in that packing list, use the following command format:

```
aradmin -u <username> [-p <password>] -x <servername>
-i <source_file> -l <packing_list.xml>
```

The `-l` option parses the XML packing list and imports all objects defined within the packing list. This option overrides other options in the same command.

Converting form views

You can convert native views to web views by using the following command formats.

To convert all native views to web views, use the following command format:

```
aradmin -u <user_name> [-p <password>] -x <server_name> -convert
```

To convert specific form views, use the following command format:

```
aradmin -u <user_name> [-p <password>] -x <server_name> -convert  
-v <view_name> -f <form_name>
```

To convert views using an XML option file to define copy, prefix, and suffix operations, use this command format:

```
aradmin -u <user_name> [-p <password>] -x <server_name> -convert  
-option <XML_file_name>
```

Obtaining AR System server configuration details

Use the `-c` command to obtain configuration details for a specified AR System server. You can send all configuration details for the specified server to a file, including information not normally displayed by BMC Remedy Administrator.

To obtain configuration information for a server, use the following command format:

```
aradmin -u <user_name> [-p <password>] -x <server_name>  
-c <file_name>
```

Synchronizing an AR System search database

Use the `-s` command to synchronize the search database.

```
aradmin -u <user_name> -x <server> -s
```

Using the BMC Remedy User CLI

You can start BMC Remedy User through the command line with the `aruser` command. This command uses the following parameters:

- `app`—The AR System application if you want to launch BMC Remedy User in application mode
- `auth`—The external authentication string
- `server`—The server to which you want to log in
- `user`—The BMC Remedy User user name
- `password`—The password for the user

For example, the syntax to start BMC Remedy User with a login name and password is:

```
Aruser /user="Demo" /password="remedy"
```

Or

```
Aruser -user="Demo" -password="remedy"
```

This would launch BMC Remedy User and automatically log the user in as “Demo” with a password of “remedy.”

To use an external authentication string, the syntax include the `auth` parameter:

```
Aruser /user="Demo" /password="remedy" /auth="<authentication string>"
```

Or

```
Aruser -user="Demo" -password="remedy" -auth="<authentication string>"
```

Using the runmacro CLI

The AR System server includes the runmacro utility, which can run a macro or export data without a GUI, as a background process. The `runmacro` utility can be run from filter or escalation workflow, or as a standalone process (that is, a Windows batch file or UNIX script). The `runmacro` utility can also be used by third-party applications to run AR System macros. Because `runmacro` functionality provides no GUI support, it can execute processes that run in the background and complete, but it cannot perform tasks such as displaying a results list of a set of records.

The `runmacro` command has two formats, which follow. The items between square brackets are optional. Enclose arguments that contain blank spaces or symbols in double quotes.

- You can use the original version of `runmacro` without the output file option (`-o`):

```
runmacro [-h <home_directory>] [-d <macro_directory>]
[{-x <server_name>} ...] { -e | -i } <macro_name>
[-p <parameter>=<value> ...] [-U <user_name>] [-P <password>]
[-Q <internal_qualification_format>]
[-q <client_tool_qualification_format>]
[-Z <internal_format_qualification_file_name>]
[-z <client_tool_format_qualification_file_name>]
[{-w | -W } <external_authentication_string>] [-a <port_number>]
[-O]
```

- You can use `runmacro` with the `-o` option to use the `arcopy` syntax, which copies the output to a file:

```
runmacro -o <output_file_name> [{-x <server>} ...] -U <user>
[-P <password>] [{ -f | -s} <form>] [-t {arx|csv|xml}]
[-Q <internal_qualification_format>]
[-q <client_tool_qualification_format>]
[-Z <internal_format_qualification_file_name>]
[-z <client_tool_format_qualification_file_name>]
[{-w | -W } <external_authentication_string>] [-a <port_number>]
```

When exporting data with attachments by using the `-o` option, the attachments folder is created in the same directory as the export file. The attachments folder name uses an integer timestamp (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 are performing 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).

The following table lists the options to `runmacro`, which might appear in any order on the command line.

Option	Description
-o	Output file name—Specifies the name of the file where you want the data stored. The file is initially truncated, and then all the data is written to the file (one data set after another).
-h	Home directory—Specifies a path to the <code><ar_home_dir></code> directory. If you do not specify the -d option, <code>runmacro</code> also looks in this directory for the <code>arcmds</code> directory that contains the macro to be 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—Specifies the directory that contains the macro if your macro is not in the <code><ar_home_dir>\arcmds</code> directory or if you do not have a <code><ar_home_dir></code> directory.
-x	Server name—Specifies the name of a server to connect to. This option might be included more than once to connect to multiple servers. Use the following format: <code>-x <server_name></code>
-e (or -i)	Macro name—Specifies the macro to be run.
-p	Parameter—Specifies a 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 quotes to cause proper interpretation of the special characters. Use the following format for each parameter specified: <code>-p <parameter>=<value></code>
-U	User name—Required login parameter that identifies the user account. The -U option must be in uppercase.

Option	Description
-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—Provides a query in AR System internal format.
-q	Client tool qualification format—Provides a regular query, for example, like you would use in the advanced search bar in BMC Remedy User.
-Z	Internal format qualification file name—Provides a file name containing the query in Remedy internal format.
-z	Client tool format qualification file name—Provides 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—Specifies the 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 <i>Configuring</i> guide.
-a	Port number—The port number to which to connect the server.
-f (or -s)	Form name—Specifies the form that will be 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 will be 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 will export <i>all</i> available regular data forms. It will not export join or external forms.
-t	Type of file to write—Specifies the file type for the output file: arx, csv, or xml. If not specified, the default of arx is used.
-0	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 CLI 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 <ar_install_dir> 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:

```
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:

- 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 new Service Request according to the macro definition.

Using the BMC Remedy Import CLI

This section discusses the BMC Remedy Import CLI. Every cross-platform CLI command opens a separate BMC Remedy Import session, executes the command, and logs out. Therefore, you must log in with every command. If BMC Remedy Import does not find the user name, BMC Remedy Import prints the usage messages and exits.

Note: On Chinese UNIX systems, CSV and ASCII files cannot be imported if they contain Date/Time fields.

When using the arimportcmd feature 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.

When moving files from Windows to UNIX systems, use FTP to ensure a stable transfer.

Running arimportcmd on a UNIX machine

When running AR System on UNIX and using arimportcmd, you must add an entry to the library path before running the command. The following examples describe how to set up the path with a Bourne shell:

AIX

```
LIBPATH=$LIBPATH:/<ARServer_install_directory>/bin  
export LIBPATH
```

HP-UX

```
SHLIB_PATH=$SHLIB_PATH:/<ARServer_install_directory>/bin  
export SHLIB_PATH
```

Solaris

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/<ARServer_install_directory>/bin  
export LD_LIBRARY_PATH
```

Importing with mapping

Importing with mapping refers to running the BMC Remedy Import CLI using a mapping created in BMC Remedy Import. The mapping must be created on Windows because that is the only environment BMC Remedy Import runs on interactively, but once created, the mapping can be used on any platform. The -m or -M option in the command line determines this mode. For more information about BMC Remedy Import mapping, see the *Configuring* guide.

You can override certain settings saved in the mapping 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.

Note: You must supply either `-M` with the fully qualified file name of the parameter, or the combination of `-d` and `-m`, where `-d` is the directory that contains the mapping file and `-m` is the name of the mapping as contained inside the mapping file, *not* the name of the mapping file. You cannot supply `-M` and the combination of `-d` and `-m` together.

Use the following format in the command line. Enclose arguments that contain blank spaces or symbols in double quotes. The items between square brackets are optional.

```
arimportcmd -u <user_name> [-p <password>] [-x <server_name>]  
[-w<external_authentication_string>][-r<rpc_number>][-a<port_number>]  
[-l <log_file_path>] [-e <duplicate_field>] [-n <suppress_filters>]  
[-t <multi_match_option>] [-o <data_file_name>]  
[-f <destination_form_name>] [-v <force_override>]  
-M <fully_qualified_mapping_file_name>
```

The following table describes available options.

Option	Description
<code>-u</code>	User name—Required login parameter that identifies the user account. Requires a <code><user_name></code> argument.
<code>-p</code>	Password—Optional login parameter that identifies the user account. The <code>-p</code> option requires a <code><password></code> argument. Omit the option if the user account has no password.
<code>-x</code>	Server name—Login parameter that specifies the server to log in to. The <code>-x</code> option requires a <code><server_name></code> argument. This option overrides the server specified in the mapping. If this option is not specified, the server name in the mapping is used.
<code>-w</code>	Authenticator—Specifies the 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 <i>Configuring</i> guide.
<code>-r</code>	RPC program number—Specifies a private server, for example, if a dedicated import server is available. If not specified, the default is the admin server 390600.

Option	Description
-a	TCP port number—Identifies the 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.
-d	Mapping directory—Required directory that contains the mapping file being referenced with the -m option.
-l	Full path name of the log file—Use this option to log details of the import execution.
-e	Duplicate field—The ID number of the field to check for duplicate data. For example, for the Short Description field, you would enter the value 8. By default, the Request ID field (field ID 1) is used when -e parameter is omitted.
-n	Suppress filters—When the system is merging entries on forms, this command instructs arimportcmd to suppress the merge filters.
-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.
-o	Data file name—Name of the file containing data to import. If specified, this option overrides the data file specified in the mapping. If not specified, the data file specified in the mapping is used.
-f	Destination form name—Name of the form to import into. If specified, this option overrides the form specified in the mapping. If not specified, the form specified in the mapping is used.
-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 Import client and invalidates the old IP address.
-i	Supplies a default value.

From the following options, use *either* a combination of `-m` and `-d`, or `-M`, to specify the mapping to be used. You cannot use both the combination of `-m` and `-d` with `-M`; they are mutually exclusive.

Option	Description
<code>-m</code>	Mapping name—Required name of the mapping file to be used. You can verify the required name by opening the mapping file and using the string contained in the first line of the file.
<code>-d</code>	Mapping directory—Required directory that contains the mapping file being referenced with the <code>-m</code> option.
<code>-M</code>	Full path name of mapping file—The required fully qualified path name of the mapping file to be used.

Importing without mapping

Importing without using a mapping refers to running the BMC Remedy Import CLI with no mapping definition to instruct how to map fields. This means that all field values will be mapped using matching field IDs.

Accordingly, only AR Export (.arx) and AR XML (.xml) file formats are supported because they are the only file formats that include field IDs.

Without a mapping, the server name, form name, and data file name *must* be specified in the command line. Mappings are built by querying the server and the data file.

Use the following format in the command line. Enclose arguments that contain blank spaces or symbols in double quotes. The items between square brackets are optional.

```
arimportcmd -u <user_name> [-p <password>] -x <server_name>
[-a <port_number>][-f <destination_form_name>] -o <data_file_name>
[-e<duplicate_field>][-n<suppress_filters>][-t<multi_match_option>][-l
<log_file_path>] [-D <duplicate_ID_option>] [-r <rpc_number>] [-v]
```

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. Requires a <user_name> argument.
-p	Password—Optional login parameter that identifies the user account. The -p option requires a <password> argument. Omit the option if the user account has no password.
-x	Server name—Login parameter that specifies the server to log in to. The -x option requires a <server_name> argument.
-a	TCP port number—Identifies the 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.
-f	<p>Destination form name or pair.</p> <p>A single name indicates that the form name in the source data file matches the form name on the server.</p> <p>To specify a pair of names, separate the form names with an equal sign, without any blank spaces around the equal sign: “<Destination_form>”=”<File_form>”.</p> <p>The destination form is the form on the server into which data will be imported. The file form is the form specified in the data file.</p> <p>Specifying pairs maps data from one form, specified in the data file, to a different form, identified on the server.</p> <p>Multiple pairs can be specified using this option multiple times, for example:</p> <ul style="list-style-type: none"> -f “Target_form_a”=”File_form_b” -f “Target_form_c”=”File_form_d” <p>If the -f option is not specified, BMC Remedy Import will attempt 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 will be imported. If no matching form is found, the data set will be ignored.</p>
-e	Duplicate field—The ID number of the field to check for duplicate data. For example, for the Short Description field, you would enter the value 8. By default, the Request ID field (field ID 1) is used when -e parameter is omitted.
-n	Suppress filters—When the system is merging entries on forms, this command instructs arimportcmd to suppress the merge filters.

Option	Description
-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.
-l	Full path name of the log file—Use this option to log details of the import execution.
-o	Data file name—Name of the file containing data to import.
-D	Duplicate ID—Defines how BMC Remedy Import processes records that contain request IDs, which duplicate those already in the form. With this option, you must include one of the following numbers: <ul style="list-style-type: none"> ■ 0: Generate new ID for all records ■ 1: Reject duplicate records ■ 2: Generate new ID for duplicate records ■ 3: Replace old record with new record (the default) ■ 4: Update old record with new record's data
-r	RPC program number—Specifies a private server, for example, if a dedicated import server is available. If not specified, the default is the admin server 390600.
-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 Import client and invalidates the old IP address. Note: This option does not apply to users with administrator permissions.
-i	Supplies a default value.

BMC Remedy Import CLI examples

With mapping

The following examples show you how you can use `arimportcmd` with mapping:

- In the following example, the server name, form name, and data file name are optional because the mapping file contains the information:

```
arimportcmd -u <user_name> -p <password> -m <mapping_name>
-d <mapping_file_directory> -l <log_file>
```

- In the following example, the server name, form name, and data file name override the names in the mapping file. When you use arimportcmd with mapping, you can override one or more of the three names.

```
arimportcmd -x <server_name> -u <user_name> -p <password>  
-m <mapping_name> -d <mapping_file_directory> -l <log_file>  
-o <data_file_path> -f <form_name>
```

Without mapping

Without mapping, 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 are working with multiple servers on the same machine, you can use -d for duplicate record handling and -a to specify a port number.

The following examples show how you can use arimportcmd without mapping:

- In the following example, minimal options are used. The *data file* 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.

```
arimportcmd -x <server_name> -u <user_name> -p <password>  
-o <data_file_path> -l <log_file>
```

- In the following example, the *form name* determines which set of data from the data file will be imported to the server. The form name on the server and the data file should match.

```
arimportcmd -x <server_name> -u <user_name> -p <password>  
-f <form_name> -o <data_file_path> -l <log_file>
```

- In the following example, an import is being attempted into the form called A on the server, but the data comes from form B in the data file.

```
arimportcmd -x <server_name> -u <user_name> -p <password>  
-f "A=B" -o <data_file_path> -l <log_file>
```


Chapter 17 XML import and export

Integration with non-AR System data is available at levels that were impossible or difficult before. Import and export utilities have been improved, allowing data and view definition files to be processed using standardized XML formats. This allows administrators to localize data easily and to share data across non-Remedy databases and applications.

This feature also allows the flexibility and convenience of connecting to legacy data systems or sources, again supporting backwards compatibility.

The following topics are provided:

- AR System objects in XML (page 250)
- AR System data in XML (page 250)
- Using XML with the AR System API (page 251)

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 will have 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. See the data structure information in the *C API Reference* guide for more information about the XML Schema definitions of AR System objects.

AR System XML definition files are used the same way as the classic .def (definition) files. When exporting objects in BMC Remedy Administrator, 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 Import. Open your XML data file by selecting AR XML (*.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* guide.

Chapter 18 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 254)
- Client and server processes (page 254)
- Implementing the Run Process action to execute another application (page 255)
- Implementing the Run Process action to retrieve data from another application (page 260)
- Run a process on the web (page 263)
- Issues and considerations (page 264)

Overview

One of the simplest ways to integrate two applications is to execute one application from within another. AR System allows 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 16, “Command-line interface (CLI),” for more information.

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 runtime 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 Asynchronously**—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 only the clients.

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.

Implementing the Run Process action to execute another application

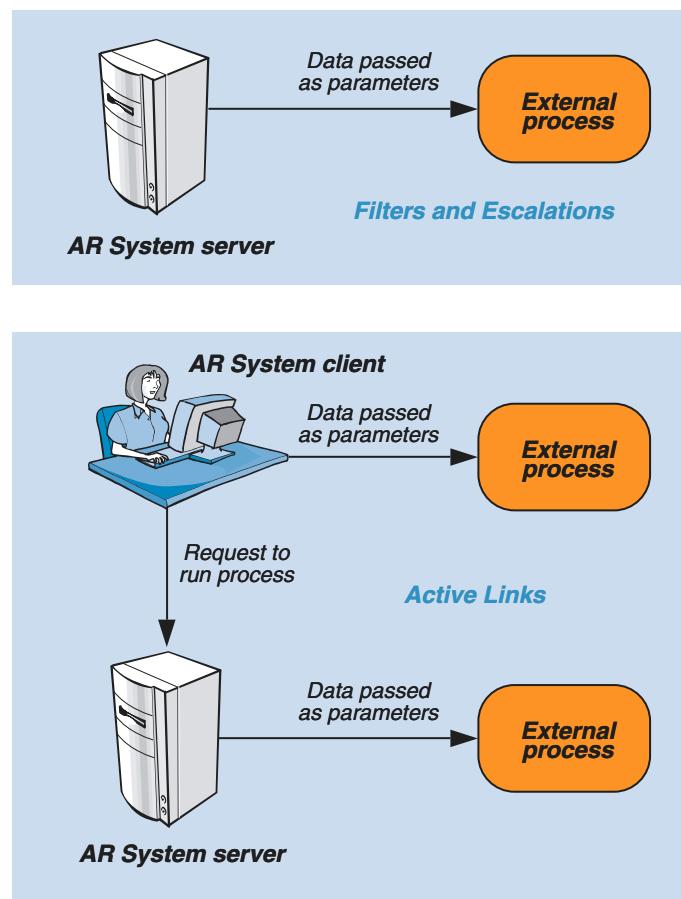
The Run Process action is used to start an external application. Depending on the function and behavior of the external application, the application can be executed through any of the following means:

- 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 runtime arguments to the application.

The Run Process action simply executes an independent process; it does not return a value to the calling program.

Figure 18-1: Executing another application



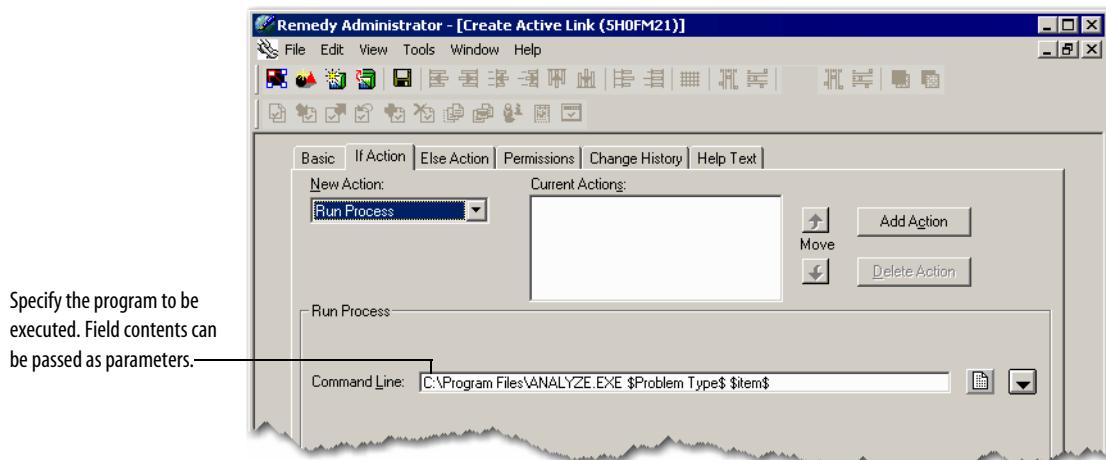
Unlike active links, which run with the access control permissions of the user, filters and escalations run with the permissions of the administrator and thus have access to all form fields. This should be taken into consideration when defining filter or escalation actions since they can have security 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 18-2. When the active link is triggered, the system will execute the command specified in the Command Line field, which will launch a related process. Note that the full path name to the executable should be specified to make sure that it will be executed correctly on the client machine. When the application is started, data from the current form can be passed as runtime arguments to the application.

Figure 18-2: Defining a Run Process action for an active link



When designing an active link that uses a Run Process action on the client, always consider the variety of client platforms that the users will be using. 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 <install_directory>\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 new 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 would be that the Reference Documents field is not empty. The active link action would be 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 who is identified in the Responsible Person field on the request. You will 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.

Implementing the Run Process action to retrieve data from another application

Another type of action, Set Fields, allows you to include workflow that automatically sets the contents of various fields from a variety of possible data sources.

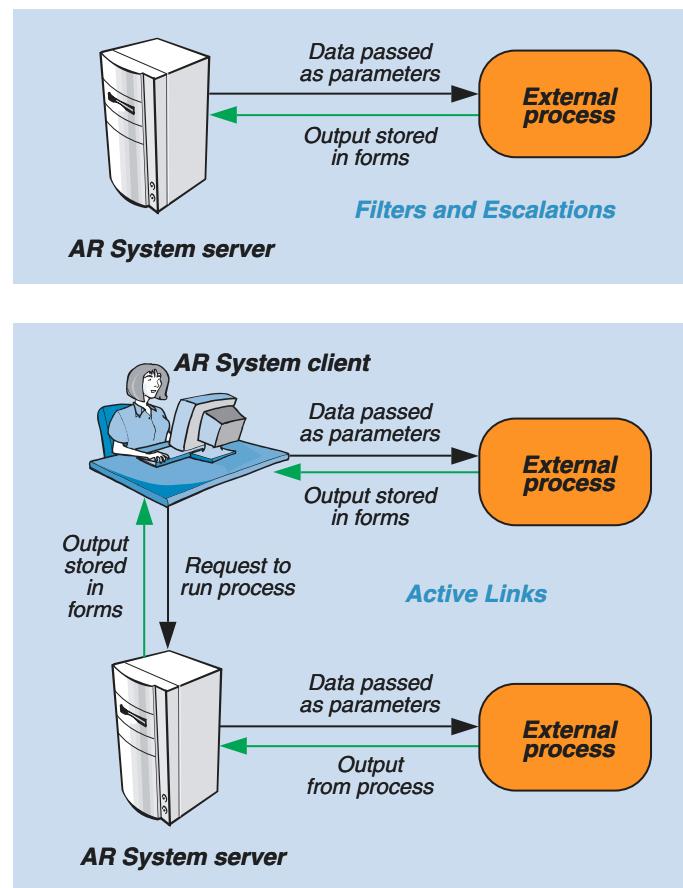
These data sources include fixed values, values read from other forms (possibly in other databases), values resulting from arithmetic operations or functions, and values returned by external applications. AR System can execute an application, and use the output of that application to set the contents of various data fields. This can be done using the \$PROCESS\$ keyword of the Set Fields action. You can only run a process that:

- Runs in a console (such as a .bat script or runmacro.exe), but not a GUI application.
- Returns 0 if successful. (In this case, stdout will be pushed to the field.)

If the process returns anything other than 0, stdout displays an error.

Again, the choice of using 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, while filters and escalations execute only on the server.

Figure 18-3: Retrieving data from another application



When an external application is run on the server, AR System will wait 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 Server Information window of BMC Remedy Administrator.

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 will be started, and BMC Remedy User will wait for it to complete. (In UNIX, the process runs in a Bourne shell.) All data returned will be 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 will be 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 the users will be using. 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 to be supported on multiple platforms, each platform will require its own active link with an appropriate qualification.

Note: You can run a process on a server by inserting @<server_name>: before the process name in an active link. For example,

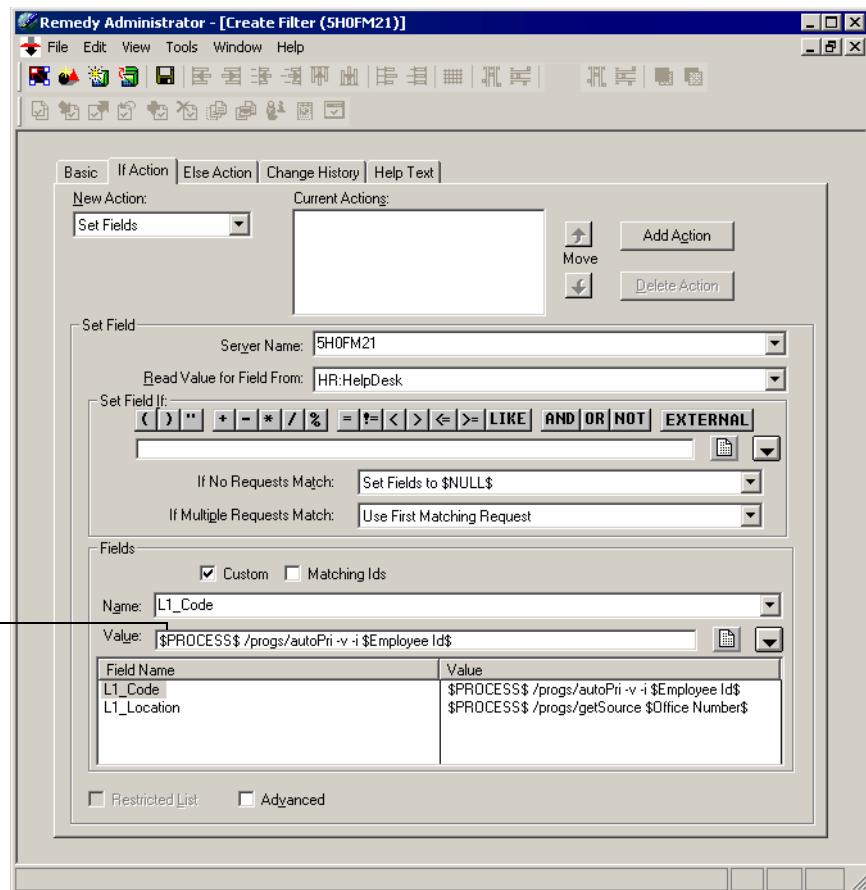
\$PROCESS\$ @ServerA: C:/temp/process.exe

If it is the current server, you can use @@ instead of @<server_name>.

An example of a Set Fields action for a filter is shown in Figure 18-4. 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.

Assuming these programs execute correctly (that is, return with an exit code of zero), their outputs will be assigned to the respective fields.

Figure 18-4: Defining a Set Fields action using \$PROCESS\$



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 will be 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 presses 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 field value has a double quote or a backslash or a new line in it, there will be a JavaScript error.
- If the word `javascript` is not at the beginning of run process action, it will say “The following specific error(s) occurred when executing ‘xx’,” but it will 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 quotes 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 of 256 characters. 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 19 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 268)
- AR System and OLE automation (page 269)
- Active links and OLE automation (page 270)
- 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 280)
- Issues and considerations (page 285)

For more information about automation in AR System, see the *C API Reference* guide. For a sample of using OLE Automation, see Appendix D, “Sample exercise: using OLE automation.”

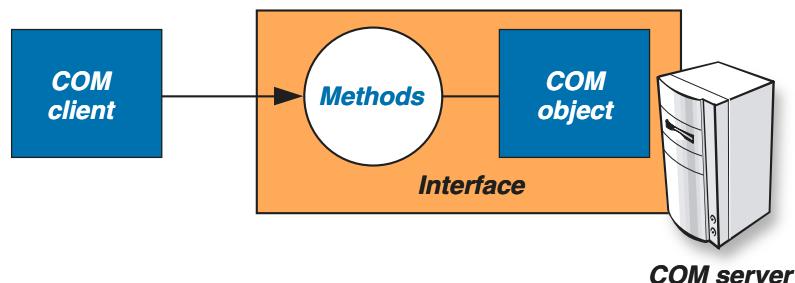
OLE overview

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 the 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.

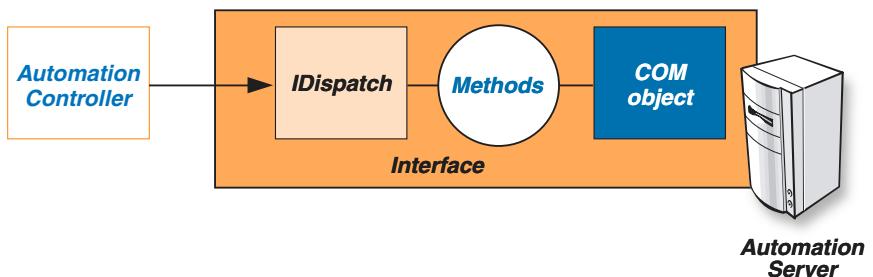
A *COM object* is software that provides its services to other software through one or more *interfaces*, each of which includes several methods. A *method* is typically a function or procedure that performs a specific action and can be called by the software that uses the COM object (the *client* of that object).

Figure 19-1: 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 19-2: OLE Automation Architecture



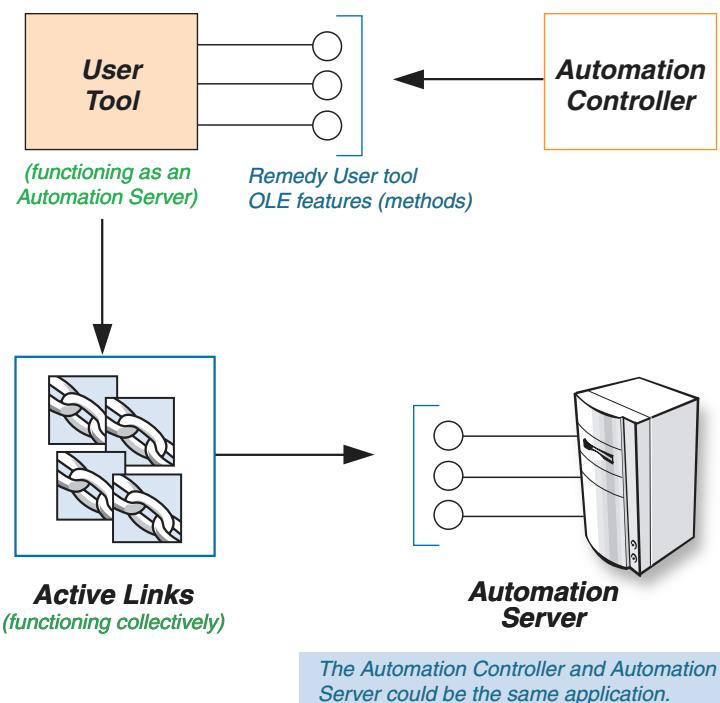
When Microsoft added Web-aware features to OLE, it changed the name to ActiveX. An *ActiveX control* is an example of an Automation server that can be controlled by BMC Remedy User. ActiveX controls are objects that can be reused by many applications; many of them can be downloaded as small programs 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 19-3: OLE Automation in AR System



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 within 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 Administrator 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.

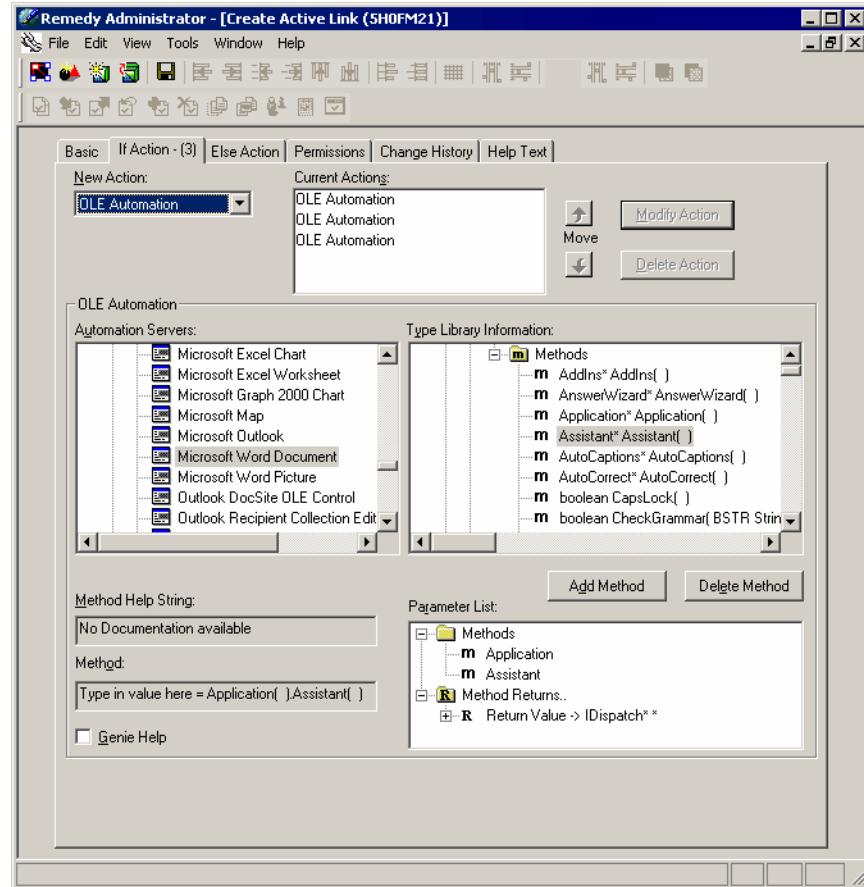
- The Automation server must have the following properties listed under its registry entry:
 - TypeLib listing the typelib IID (interface identifier) or Programmable.
 - InProcServer or LocalServer entry.
- When the servers have any one of the following properties, they are not displayed in BMC Remedy Administrator:
 - If the in-process server (InProc) is a system component and not an application OLE server, it will be skipped.
 - If the Automation server is an OLE 1.0 control, it will be 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 will execute must contain the same server or control where the active link was defined. If you are building your own Automation server or control, you must register it on the user machine using the Regsvr32 utility. See the Microsoft's OLE, COM, and ActiveX documentation for more information about how to register servers and controls.

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 tabs for creating OLE Automation active links.

Figure 19-4: OLE Automation active link action

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 Administrator 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:

<return-type><method-name>(parameters).

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 Administrator 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 have chosen a server, an object, and a method to invoke, click the AddMethod button in the Active Link dialog box to include the method in the call sequence. The Parameter List pane displays the method tree and associated parameters. The method node along with return types are also displayed. If the return type of a method is void, then 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 illustrates how to open Excel 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 will terminate 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 will require 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 are nesting, the return type of the method to be added and the parameter type should match. BMC Remedy Administrator will validate the type compatibility when you are nesting 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, the BMC Remedy Administrator tool 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 typed 1234 as the parameter value. As soon as the focus moves off of the parameter value node, denoting completion of the action, BMC Remedy Administrator 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 had typed in `Test` as the parameter value, the variant change operation would have failed with the appropriate error message displayed.

If a parameter is a pointer type variable and the user types in a value, BMC Remedy Administrator displays an error message. In this case, the user is only allowed to add methods that have the same pointer type as the return value.

A parameter that is an object pointer will not accept a typed-in value but will accept a method supported in the ActiveX control that will return that object.

Methods of other interfaces can be selected, but make sure the return type of the method to be added and the parameter type are the same.

A right-mouse click on the parameter value node will display 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 Administrator will not be able to determine the underlying type of the variant at design time. However, BMC Remedy User will perform validation at run time and will display any error messages if a type mismatch should occur.

Working with method return types

Values cannot be entered directly at the return node. However, AR System fields can be used to accept the return value of the method. Using the right-mouse button, a menu list will appear, and the user 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 both keywords and fields grayed out, 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 choose the `Application*Application()` method. This method will return an application object.
- 3 Select the `_Application Object` and choose the `Void Visible` method. Any method can be nested from the application object.

When a method is added to the return type, BMC Remedy Administrator will use the following rules to validate the method:

- BMC Remedy Administrator 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 Administrator cannot predict which object the `IDispatch` interface represents at design time. (BMC Remedy Administrator only validates that a value cannot be directly entered like an ordinary parameter.)
- If the return type is `VARIANT`, no validation is performed because BMC Remedy Administrator 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 Administrator will allow the nesting of methods as `SetDispatch(GetBSTR())` because BMC Remedy Administrator is not aware that the `VARIANT` from `GetBSTR` will contain 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 will act on the same server. Once the active link has finished 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 class module defined becomes an interface object. In Visual Basic, choose the Class property and change the ClassName.
- Provide a recognizable name to the Project Name. In Visual Basic, choose the Project Menu > Properties > ProjectName.
- Make sure the .dll is created and registered in the system by using the Regsvr32 utility, for example, `regsvr32 <.dll Name>`.

When you are using an ActiveX control as an OLE server, you perform the same procedure described in Appendix D, “Sample exercise: using OLE automation,” on page 449; however, 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* guide.

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 within AR System. Like 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 Administrator.

A sample exercise is outlined in Appendix D, “Sample exercise: using OLE automation.” This example uses OLE automation to create a form and active link to open Microsoft Word, spell check text in a field in that form, and return the corrected text to the field.

Note: If you are designing 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. See the *Workflow Objects* guide for more information about building qualifications.

► To define the OLE automation active link action

- 1 With the server window open, choose File > New Server Object.

The New Server Object dialog box appears.

- 2 Select Active Link from the New Server Object field.

- 3 Click OK.

The Create Active Link dialog box appears.

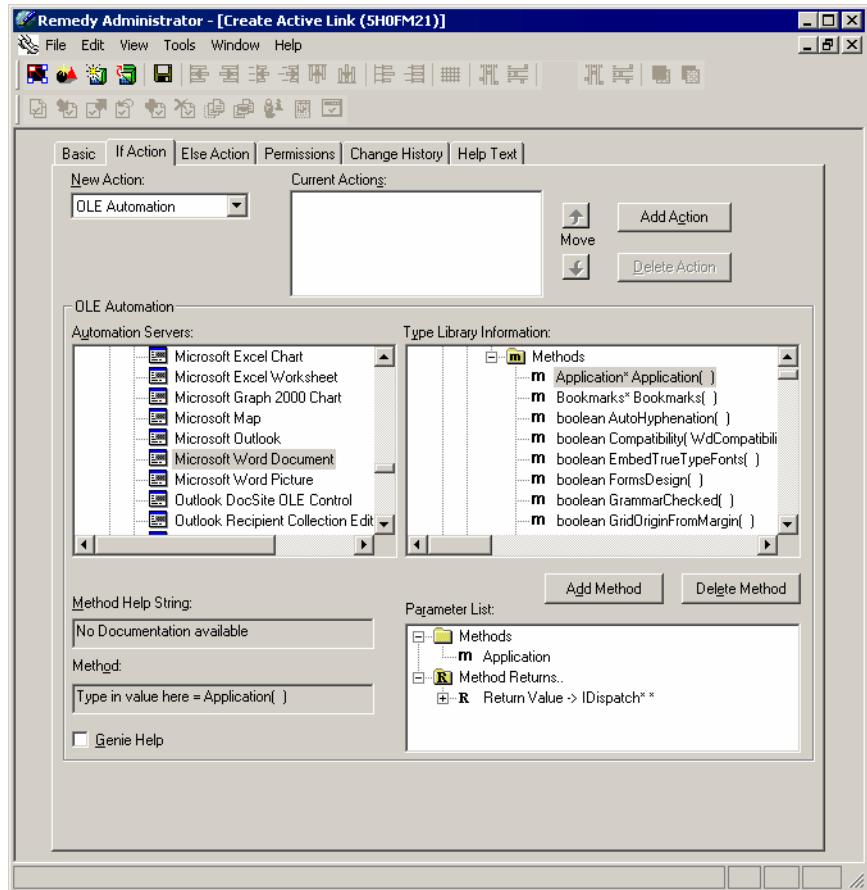
- 4 Set up the conditions in the Basic tab.

- 5 Click the If Action tab, or click the Else Action tab.

- 6 From the New Action list, select OLE Automation.

The fields required to define the OLE Automation action appear. The following figure shows these fields and an example of how an OLE Automation active link action might look after you complete the remaining steps in this procedure.

Figure 19-5: OLE Automation active link action



- 7 If you want animated Genie assistance as you complete the rest of the procedure, select the Genie Help check box.
- 8 From the 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 Administrator also exist on their computers.

- 9 Double-click the appropriate server or control.

The list of objects defined in the OLE server is displayed under the Type library tree hierarchy.

- 10 From the Type Library Information list, select the Objects folder.

A list of objects defined for the selected server or control appears. You can also select the Enumerators, Structures, Objects, Aliases, or Unions directories to view display-only reference information for the selected server or control. If the selected server or control has at least one of these items defined for it, you will only be able to open the Enumerators, Structures, Objects, Aliases, or Unions directories.

- 11 From the Objects folder, select the appropriate object.

A Methods folder appears that contains all of the methods for the selected object.

- 12 Open the Methods folder, select the appropriate method, and then click Add Method.

The selected method is displayed as an equation in the Method field. At the same time, the method is added to the Method folder in the Parameter List as a tree control. You can use the tree control to define variables that make up the method parameters and return value. As you define each method parameter and the return value through the Parameter List tree control, the Method field displays the results of each selection you make.

When you look at the method equation in the Method field, each item that you can specify has the place holder ?Type in value here. If ?Type in value here is not in the method equation where you would expect to find the return value or a method parameter, it means that a return value or method parameters do not exist for the selected method.

If an OLE method returns an object, you can nest the methods of the returned object. That is, if when you call a method on object A, object B is returned, you can nest these methods using the syntax:

Return Value = A.B(<parameter 1>, <parameter 2>, ...).

- 13 To define the method parameters, click the plus sign to the left of the method name, click the plus sign to the left of Parameters, and then click the plus sign to the left of the appropriate parameter name.

The words ?Type in value here appear below the selected parameter and represents the item that you want to use as the parameter when the method is called. The parameter value must be of the data type specified in the parameter name. You can define the parameter in one of the following ways:

- Select ?Type in value here, click again to activate edit mode, and then type in the appropriate parameter value.
- Right-click ?Type in value here, and choose from a list of field values or keywords.
- Click ?Type in value here, select a method from the Type Library Information list that has a return value that is compatible with the parameter data type, and then click Add Method.
Your selection replaces the ?Type in value here placeholder.
- Optional parameters are displayed within brackets ([]). BMC Remedy User will substitute default values for the optional parameters when BMC Remedy User is run if you leave the parameters blank.

Repeat step 13 until every parameter for the selected method has been defined. If you set a parameter with a method, as described in step 12, and the method you choose has parameters, you must define these parameters as well.

- 14 To define the method return value, click the plus sign to the left of the Method Returns, and click the plus sign to the left of Return Value.

The message ?Type in value here appears and represents the variable that you want to use as the return value when the method is called.

- 15 Right-click ?Type in value here, and choose from a list of field names.

The return value must be of the data type specified in Return Value. Your selection replaces the ?Type in value here placeholder.

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.

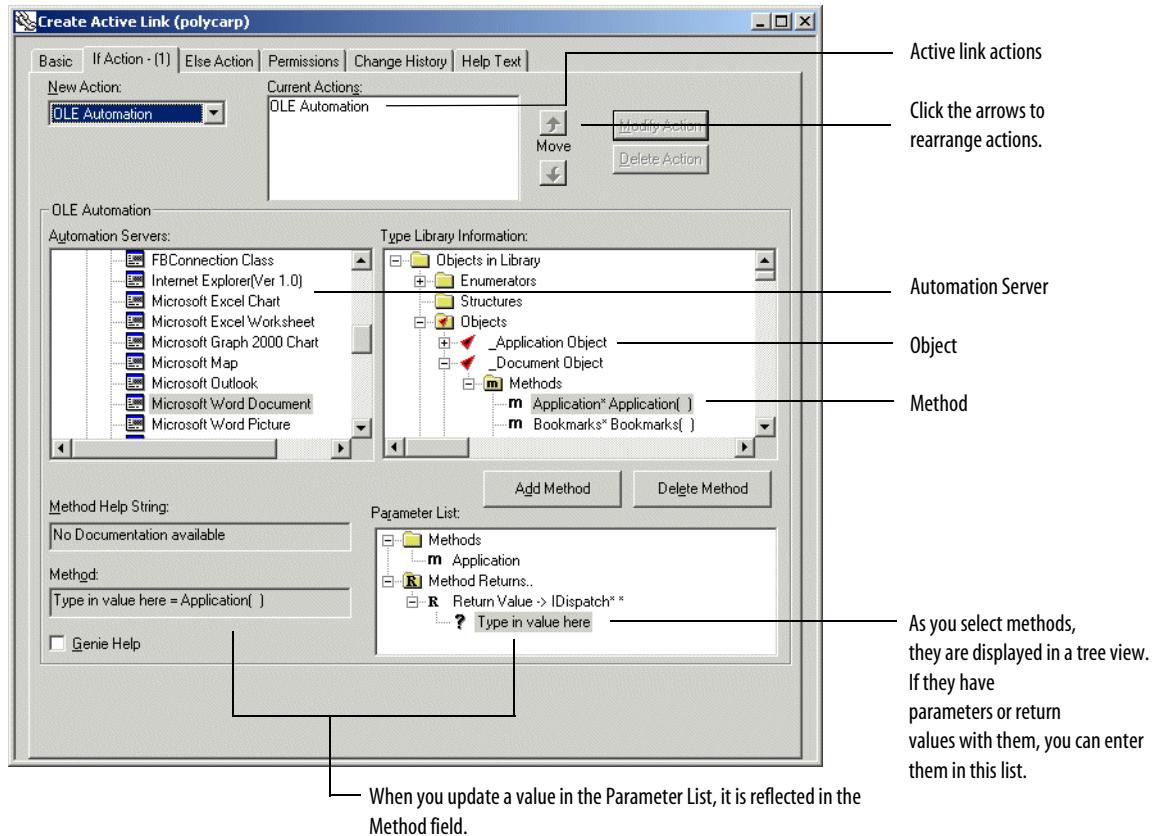
- 16 To delete a method, select the method in the Parameter list tree control, and click Delete Method.

- 17 Click Add Action (or click Modify Action).

The OLE Automation action appears in the Current Actions list.

The following figure shows an example of an active link that is already created, and explains the different sections of the If Action tab.

Figure 19-6: OLE automation active link



The Type Library Information list includes enumerators, structures, objects, aliases, and unions. OLE automation servers define these special data types, which can be used as parameters or return types in the method calls defined in the objects. Some of these data types are used for display purposes only and are not useful to the OLE automation author. However, objects contain methods defined within them, and you will be using these to create an OLE automation active link action.

When you expand the objects listed in the Type Library Information list, all the methods for each object are displayed in a tree view. The type library information is identified by using the following order:

- Object: Red icon
- Method: m
- Parameter: p
- Return Value: R

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://supportweb.remedy.com/>. 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 only works 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 20 Dynamic data exchange (DDE)

DDE can be used for AR System client integration and, indirectly, for server integration.

The following topics are provided:

- Overview (page 288)
- Configuring your system to use DDE with AR System (page 290)
- Using active links with DDE (page 294)
- Using BMC Remedy User reporting with DDE (page 299)
- Triggering AR System using a DDE execute from an external application (page 302)
- Using AR System with DDE, third-party applications and macros (page 303)
- Examples (page 307)
- Issues and considerations (page 318)

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 will establish 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 will 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 be sent to the DDE application. The data can be a literal value or a data field value.

See the Microsoft Windows documentation for more information about DDE.

Methods of integration

You can use DDE to integrate AR System with third-party applications in the following ways:

- Using the DDE active link action to send data to or execute a command in another Windows application. For more information, see “Using the DDE active link action” on page 294.
- Using the DDE active link keyword to request data from another Windows application. For more information, see “Using the DDE active link keyword” on page 298.

In BMC Remedy Administrator, you specify a DDE command and a trigger action for an active link. Users activate the active link from the toolbar or through actions in BMC Remedy User.

- Using DDE and BMC Remedy User to send a report to another Windows application and cause the application containing the AR System report to open. For more information, see “Using BMC Remedy User reporting with DDE” on page 299.
- Using a third-party application and DDE to launch BMC Remedy User and execute a macro. For more information, see “Using AR System with DDE, third-party applications and macros” on page 303

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 accomplish this.

The `dde.ini` file must be located in the `ARHOME` directory for the AR System client. The default location for the `ARHOME` directory on an 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 = <Path_to_application>
Application = <DDE_application_name>excel
Topic = <Topic_name>
Format = <Format_name>
XFRDATA = <Transfer_mechanism>
Command1 = [<DDE_Application_command1>]
.
.
.
Commandn = [<DDE_Application_commandn>]
```

The parameters in the `dde.ini` file are as follows:

Parameter	Definition
SectionTag	<p>Identifies a portion of the <code>dde.ini</code> 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 <code>dde.ini</code> file. Use descriptive names to help you remember what each section is for. Names are not case-sensitive and cannot contain spaces.</p> <p>This value is used when the Report To Application button on the reporting facility window is selected to identify which section in the <code>dde.ini</code> file should be used to control the data transfer.</p>
Path	Defines the full execution path name to the DDE server application. If the application is not already running, it will be started 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, <code>excel</code> is the DDE server name for Excel and <code>winword</code> is the DDE server name for Word.
Topic	Controls how data is exchanged for some DDE server applications. For example, when you are executing commands, functions, or macros for Excel, the Topic should be <code>system</code> .
Format	<p>Defines which column separator is to be used by AR System when multiple fields in a data record are output. This overrides any format specification in the report definition. There are four options:</p> <ul style="list-style-type: none"> ■ 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 <code>CharsPerLine</code>. If <code>CharsPerLine</code> is not specified, the default is 80. ■ Current – uses the format and page setup defined in the application.

Parameter	Definition
XFRDATA	<p>Defines the actual data transfer mechanism. The two options are:</p> <ul style="list-style-type: none"> ■ 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 <i>n</i>	<p>Define the commands that you want to send to the DDE server application to tell it what to do with the data that has been 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 on-line Help system. In the <code>dde.ini</code> file, each command specified is enclosed in square brackets.</p>

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 where %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 don't 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 <NN> is the new DDE time-out setting in seconds and 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

There are two ways you can use active links with DDE:

- You can 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.
- You can 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 are designing an active link for a form that is also used by clients on platforms other than Windows, 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. See the *Workflow Objects* guide for more information about building qualifications.

► To define the DDE active link action

- 1 Open the server window and select File > New Server Object.
The New Server Object dialog box appears.
- 2 Select Active Link from the New Server Object field.

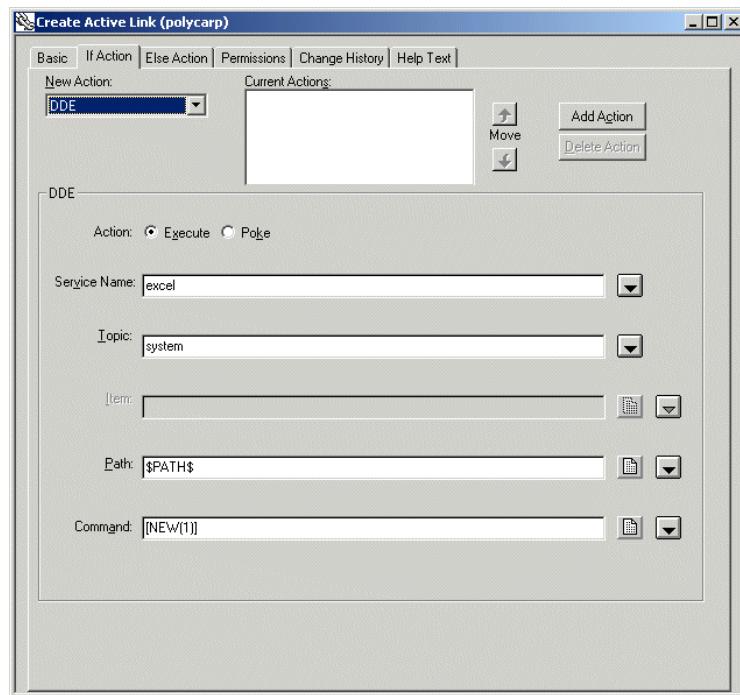
- 3 Click OK.

The Create Active Link dialog box appears.

- 4 Set up the conditions in the Basic tab.
5 Click the If Action tab or the Else Action tab.
6 From the New Action list, select 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 20-1: DDE active link action



- 7 From the Action options, select a DDE action type.

DDE action	Description
Execute	<p>Use the DDE Execute action to send a request to the DDE application to execute the commands in the Command field.</p> <p>The DDE Execute action uses these parameters:</p> <ul style="list-style-type: none">■ Service name■ Topic■ Path■ Command <p>For information about these parameters, see “DDE parameters used by AR System” on page 288.</p>
Poke	<p>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.</p> <p>The DDE Poke action uses these parameters:</p> <ul style="list-style-type: none">■ Service name■ Topic■ Item■ Path■ Command <p>For information about these parameters, see “DDE parameters used by AR System” on page 288.</p>

- 8 In the Service Name field, enter the unique ID of the DDE application.

You can use the Service Name field menu button to insert fields from the current form and append them to the service name.

- 9 In the Topic field, enter the DDE topic.

You can use the Topic field menu button to insert fields from the current form.

- 10 In the Item field, enter the DDE item (if applicable).

The Item field is enabled only if the DDE Action type is Poke. You can use the Item field menu button to insert fields from the current form.

- 11 In the Path field, enter the location of the service.

You can use the Path menu button to insert fields from the current form.

- 12** In the Command field, enter the command that you want to execute.

You must enter a value in this field. You can type the command or you can build it using the Command list to insert fields from the current form and keywords.

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.

- 13** Click Add Action (or click Modify Action).

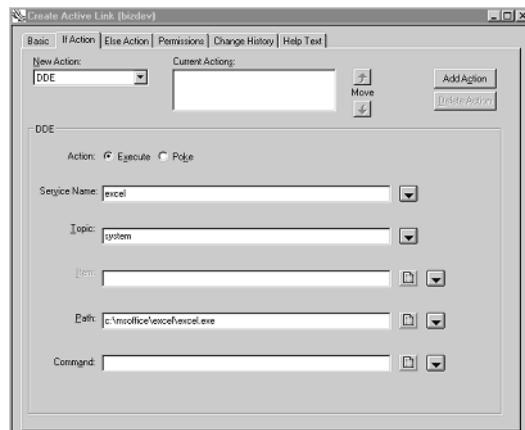
The DDE action appears in the Current Actions list.

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

In an AR System active link workflow definition, one of the possible actions is to execute a DDE command. The DDE action supports both the DDE Execute and DDE Poke functions. DDE Execute causes the DDE server application to execute some process or command. DDE Poke sends a piece of data from the DDE client to the DDE server. Figure 45 shows the form for defining an active link DDE action.

Figure 20-2: AR System Active Link DDE Action Definition



An example of a DDE Execute would be to direct Excel to start up and run a particular Excel macro. An example of a DDE Poke would be to 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

You use the Set Fields active link action to set the value of a field. You use the \$DDE\$ keyword with a Set Fields action to cause a DDE Request to be made to another application to get 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 returned by the DDE request 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:

```
$DDE$ <Service Name>;<Topic>;<Path>[;<Item>]
```

For more information about the parameters you use for the \$DDE\$ keyword, see “DDE parameters used by AR System” on page 288.

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
```

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 can send an AR System report to another Windows application, you need to perform the following tasks:

- Step 1** If necessary, create a new or modify an existing `dde.ini` file. See “Working with your `dde.ini` file” on page 290.
- 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.

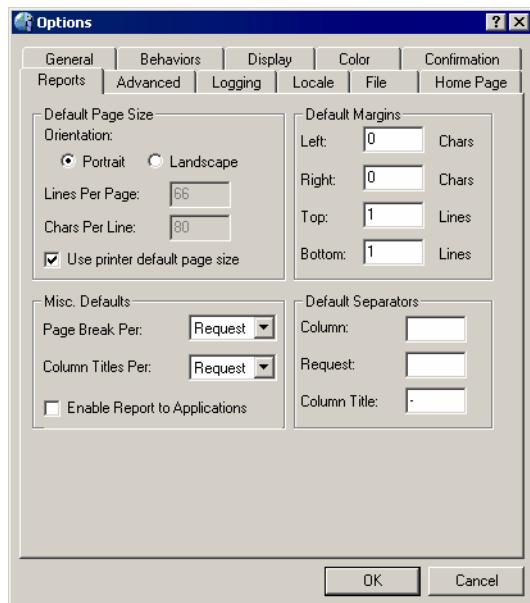
Configuring BMC Remedy User to pass report data

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.

This will enable data to be passed from the BMC Remedy User tool to other Windows applications using DDE.

Figure 20-3: BMC Remedy User options window for reporting

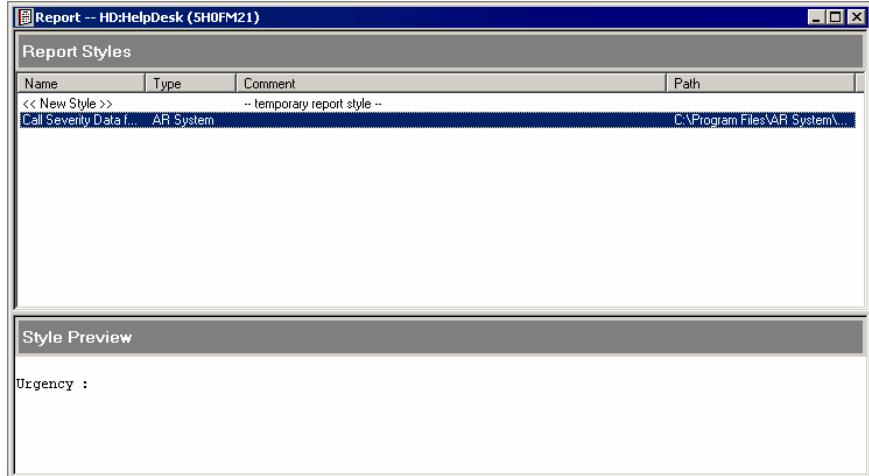
Creating a report for DDE export

You use an AR System report when you want to export large amounts of data using DDE.

► To create a report for DDE export

- 1 Open the form that contains the data you want to export.
- 2 Choose Tools > Reporting.

The Report window appears.

Figure 20-4: BMC Remedy User report window

- 3** Create a new report or select an existing report from the list.

Make sure the report you create contains the data you want 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 290.

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.

Triggering AR System using a DDE execute from an external application

Another Windows application can trigger BMC Remedy User as a DDE server. BMC Remedy User supports only one DDE command, RunMacro, which causes the 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 is issued by the DDE client to AR System.

```
[RunMacro(<MacroPath>,<MacroName>,<Parameter1Name=ParameterValue>,
<Parameter2Name=ParameterValue>,...)]
```

Parameter	Description
<MacroPath>	The fully qualified path to the directory on the PC where AR System macros are stored (for example, <code>c:\home\arcmds</code>).
<MacroName>	The complete name of the macro as defined in AR System (not the name of the file in the MacroPath directory).
<ParameterXName>	The name of a parameter that was recorded into the macro
<ParameterXValue>	The value to be substituted.

None of these parameters should contain a comma.

Note: There are no spaces in the RunMacro command statement, and 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=<path_to_aruser>\aruser.exe DDEcall <path_to_aruser>
```

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 and the RunMacro DDE function. DoExecMacro enables you to use DDE to run macros in AR System through third-party applications. The RunMacro function 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, (PFCNDCALLBACK)  
            lpDdeProc, DDE_INIT_FLAGS, NULL) == DMLERR_NO_ERROR){  
            Hssize();  
            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);
}

UnHszsize();

} // DoDDEUnInit()

/* Hszsize -- This creates often used global hszs from standard global
strings.
It also fills the hsz fields of the topic and item tables.
*/
static void Hszsize(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);

} // Hszsize()

/* UnHszsize -- This destroys often used global hszs from standard
global strings.
*/
static void UnHszsize(void)
{
DdeFreeStringHandle(idInst, hszServerName);

DdeFreeStringHandle(idInst, hszExecMacroTopic);

} // UnHszsize()

/* 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 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
DDETernate 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$
DDETernate 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

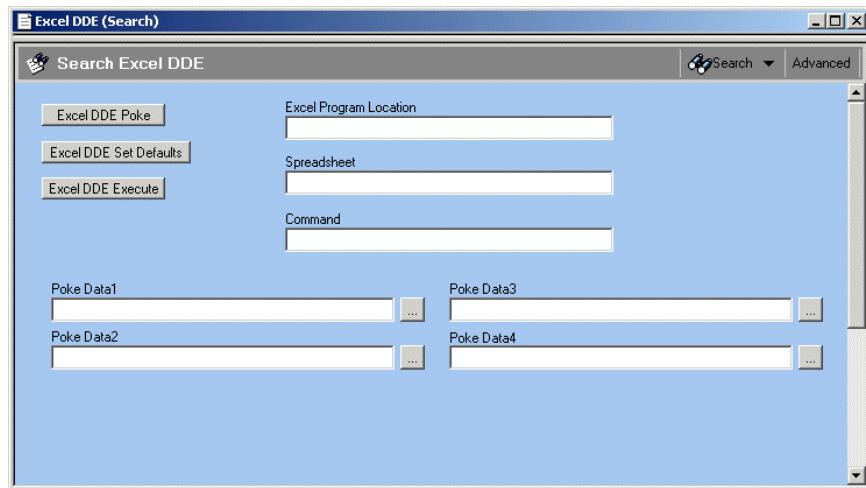
This section contains examples of integrating AR System with DDE applications.

Integrating with Microsoft Excel

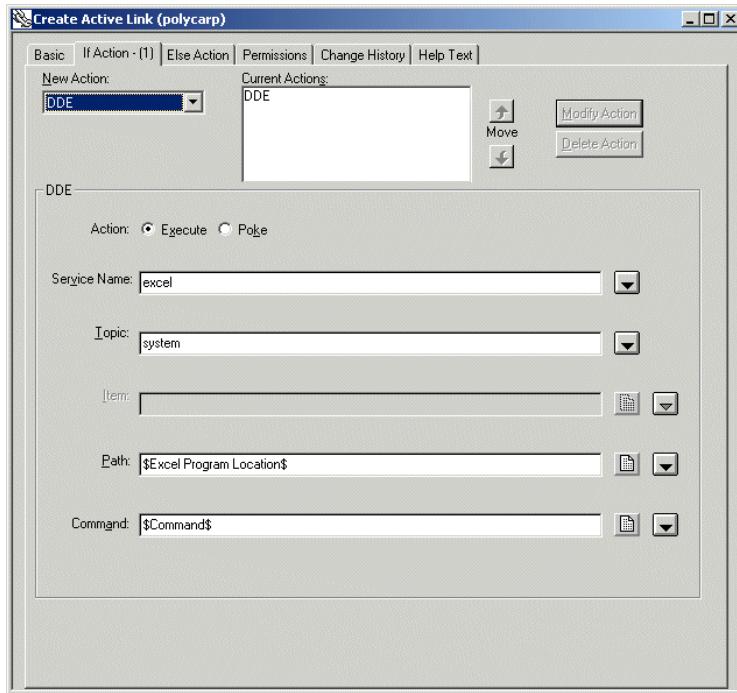
The following example includes two active links that use DDE to integrate Microsoft Excel with AR System. Using these active links, a user can open and populate a spreadsheet in Microsoft Excel through an AR System form.

The following figure shows a sample 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 20-5: 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. To create this active link, the administrator uses the If Action tab of the Active Link dialog box, as shown in Figure 20-6 on page 309.

Figure 20-6: Sample active link—excel DDE execute

- The New 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.

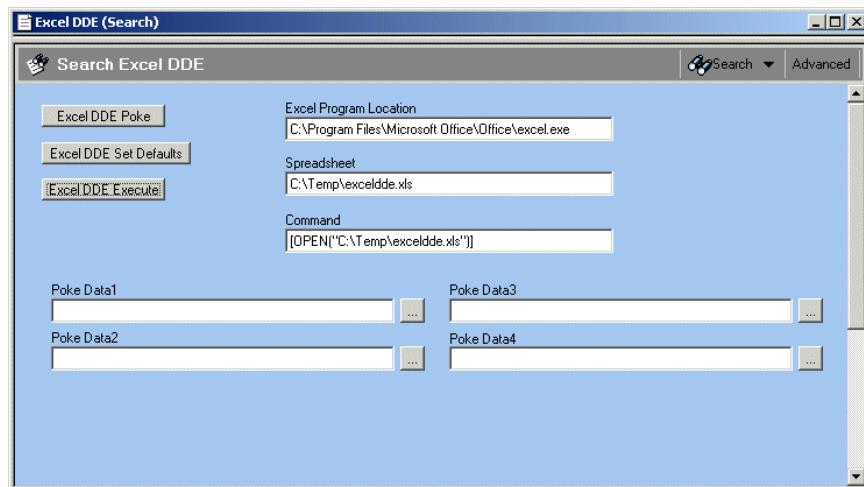
The Command field in the sample form must have a value for this active link to activate. The qualification is as follows:

```
'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.

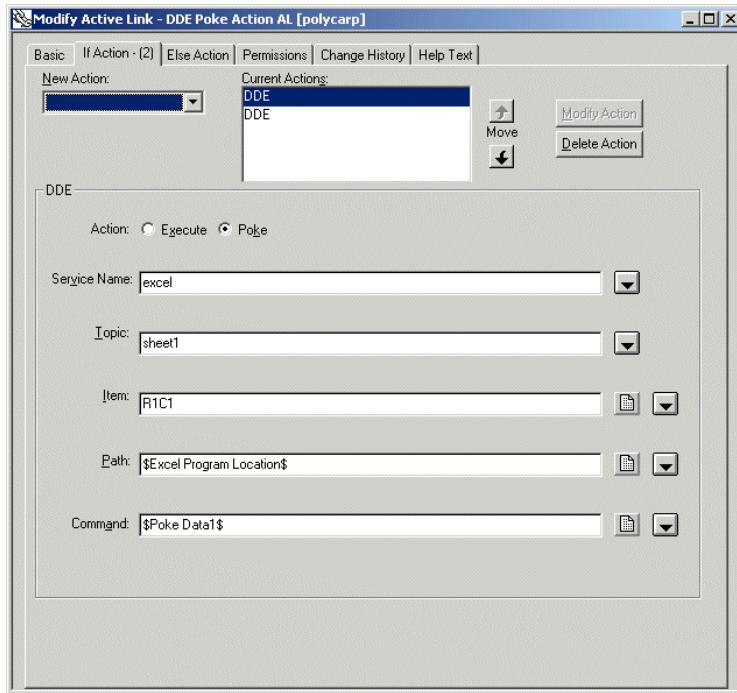
As shown in Figure 20-7, a user's Microsoft Excel executable is located at C:\Program Files\Microsoft Office\Office\excel.exe. The particular spreadsheet to be opened is located at C:\Temp\exceledde.xls. The OPEN command in the Command field opens the spreadsheet.

Figure 20-7: Sample DDE form with sample data



When the user clicks Excel DDE Execute, Microsoft Excel opens and displays the exceledde.xls spreadsheet.

When the user clicks Excel DDE Poke, the second active link is executed and a specified spreadsheet is populated. To create this active link, the administrator uses the If Action tab of the Active Link dialog box, as shown in Figure 20-8 on page 311.

Figure 20-8: Sample active link—excel DDE poke

- The Action field instructs the active link to send data to some location.
- The Item field indicates the item to which data will be 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.

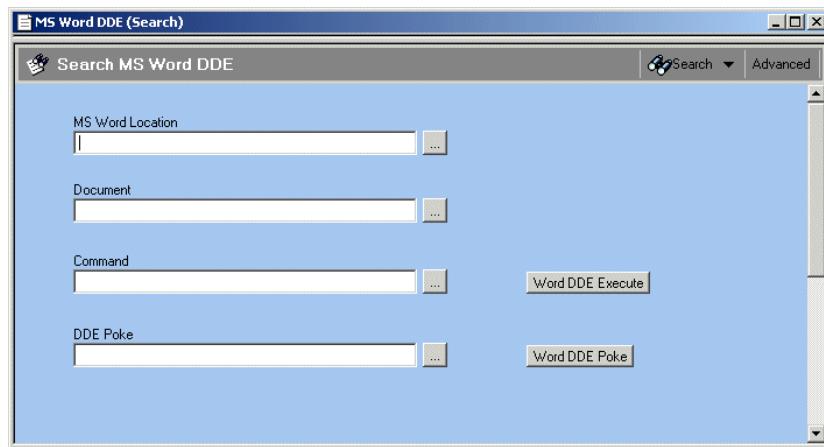
When a user supplies data to the three Poke Data fields in the sample form and clicks Excel DDE Poke, the data in the Poke Data fields are sent to cells in the spreadsheet specified in the Spreadsheet field.

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.

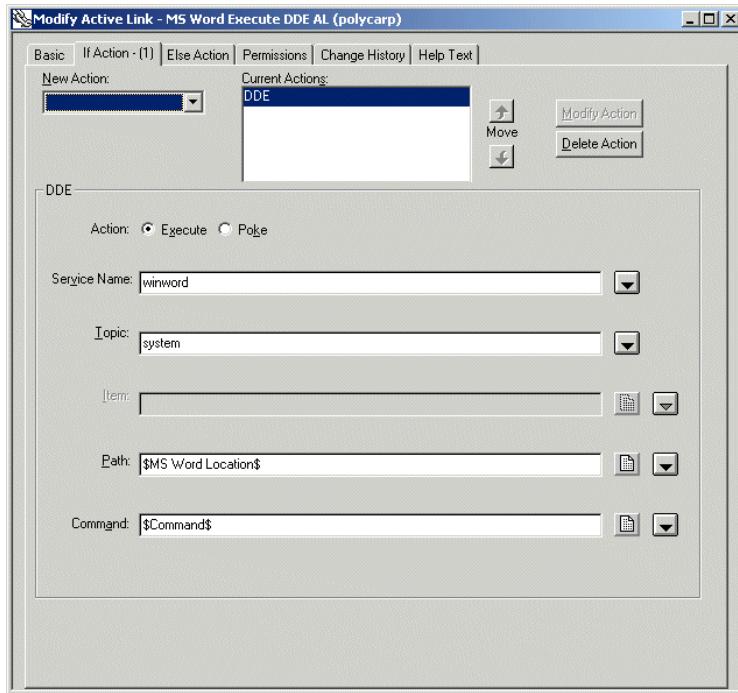
The following figure shows a sample form having active links that work with Microsoft Word. The two buttons on the right of the form execute the active links that open Microsoft Word and write to a document. The active links reference the fields to determine where Microsoft Word is installed, which document to open, and what data to send to that document.

Figure 20-9: 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. To create this active link, the administrator uses the If Action tab of the active link dialog box, as shown in Figure 20-10 on page 313.

Figure 20-10: Sample active link—word DDE execute



- 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.

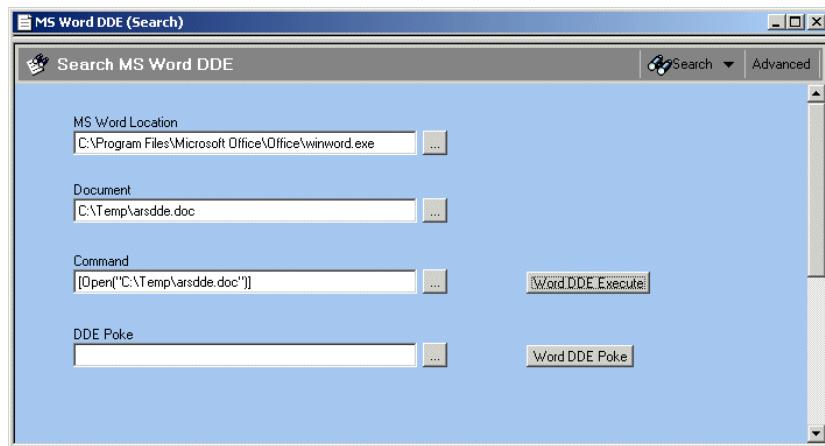
The Command field in the sample form must have a value for this active link to activate. The qualification is as follows:

```
'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.

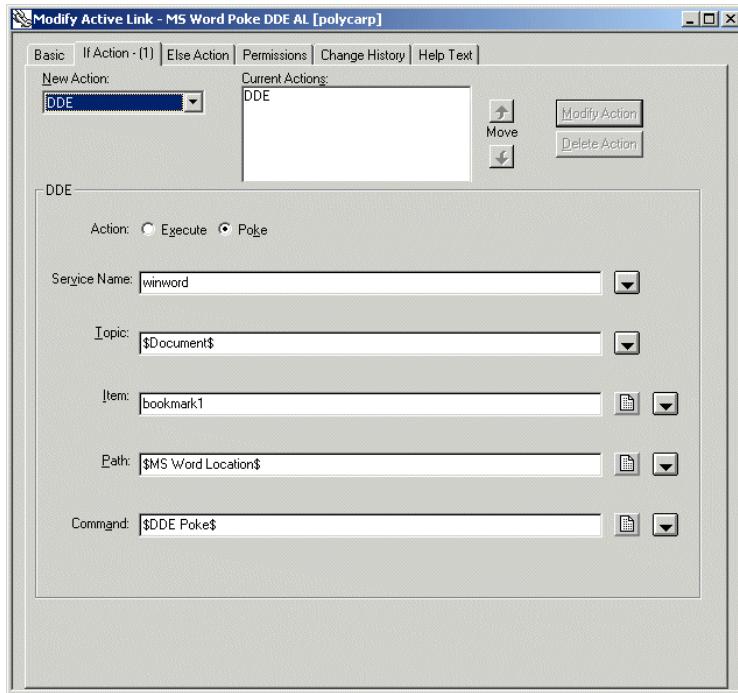
As shown in the following figure, a user's Microsoft Word executable is located at C:\Program Files\Microsoft Office\Office\winword.exe. The particular document to be opened is located at C:\Temp\arsdde.doc. This document will be opened by the OPEN command in the Command field.

Figure 20-11: Sample DDE form with sample data



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. To create this active link, the administrator uses the If Action tab of the active link dialog box, as shown in the following figure.

Figure 20-12: Sample active link—word DDE poke

- The Action field instructs the active link to send data to some location.
- The Item field indicates the item to which data will be 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 bookmark 1 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.

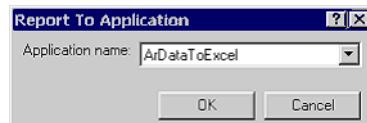
Using DDE to pass data to Excel for graphing

Assume that you have an Excel spreadsheet with a macro recorded called ChartARSystemData2Col that generates a multi-line bar chart. First, you would create a dde.ini file that controls the transfer of data from AR System to Excel as shown in the following code sample. Then you would need to define an AR System report that extracts the data that you want to graph with Excel.

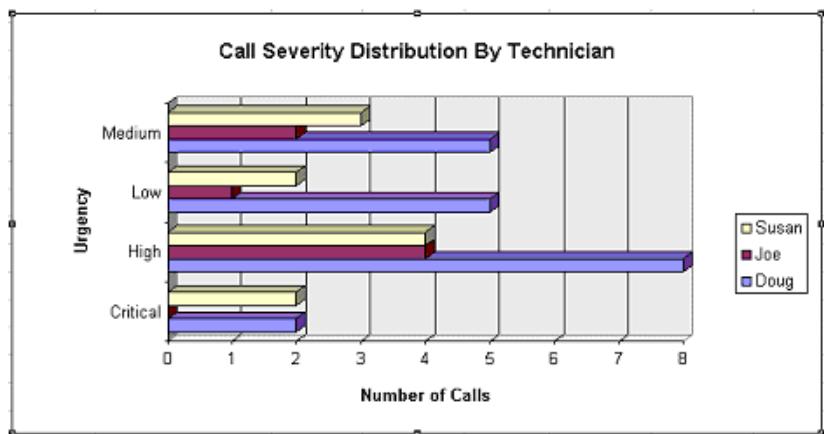
```
[ArDataToExcel]
;This section uses a temp file for temporary storage
Path = c:\program~1\micros~1\office\excel.exe
Application = excel
Topic = system
Format = TAB
XFRDATA = File
Command1 = [OPEN("%f")]
Command2 = [RUN("PERSONAL.XLS!ChartARSystemData2Col")]
```

In BMC Remedy User, choosing Report > Export to > Application causes the Report To Application dialog box to appear. The application name is the section tag from the dde.ini file.

Figure 20-13: BMC Remedy User tool Report To Application dialog box



The data is sent to Excel, an Excel macro is run (PERSONAL.XLS!ChartARSystemData2Col), and a graph is generated, as shown in Figure 20-14.

Figure 20-14: Call Distribution by Support Technician

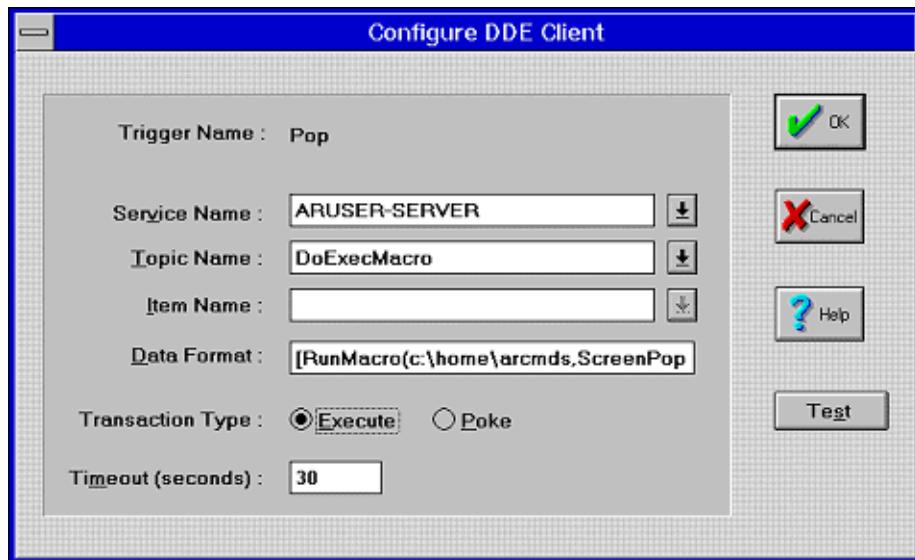
Integrating with CTI middleware application

FastCall is a Windows application that provides a link between telephony equipment and desktop applications, such as BMC Remedy User. When a call is routed by the telephone switch to a particular desktop, FastCall can trigger BMC Remedy User to open AR System windows based on parameters provided by the switch.

In this case, FastCall is the DDE client and aruser is the DDE server. Figure 20-15 shows the FastCall configuration screen. The Data Format field contains the DDE action to be executed, consisting of the following elements:

- RunMacro, which is the only DDE command supported by AR System.
- c:\home\arcmds is the path to the directory where AR System macros are stored on this PC.
- ScreenPop1 is the name of a macro that opens various AR System windows.
- PhoneNumber is the name of the AR System macro substitution parameter recorded into the ScreenPop1 macro.
- %C is a FastCall parameter that passes the caller's telephone number from the telephony switch (%D provides the called number and %I passes up to 16 characters of caller input data).

Figure 20-15: FastCall DDE Client Configuration Screen



[RunMacro(c:\home\arcmds,ScreenPop1,PhoneNumber=%C)]

BMC Remedy User will open with the caller's phone number (and other fields if they are specified) filled in.

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 21 Simple network management protocol

You can use the Simple Network Management Protocol (SNMP) to monitor AR System using the BMC Remedy SNMP agent. This section describes how to configure BMC Remedy SNMP agent if you did not configure it during the installation process.

The following topics are provided:

- Overview (page 320)
- BMC Remedy SNMP agent functions (page 321)
- Sending traps (page 323)
- SNMP configuration (page 324)
- The arsnmpd configuration file (page 325)
- The snmpd configuration file (page 331)
- The armonitor file (page 332)
- Starting the Remedy SNMP Agent (page 333)
- Stopping the Remedy SNMP Agent (page 333)
- Troubleshooting (page 334)

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 the BMC Remedy SNMP agent. You must configure the 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 the BMC Remedy SNMP Agent to monitor the 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)

The 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)

The 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://supportweb.remedy.com>.

BMC Remedy SNMP agent functions

You can use the 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

The 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

The BMC Remedy SNMP Agent monitors the state of the 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 the AR System is running; a value of 2 indicates that the AR System is down.

The managed object `arsState` is also writable, so the value of `arsState` can be changed by way of an SNMP set operation (provided the proper user name or community string is supplied). Changing the value of `arsState` from 1 to 2 will be interpreted by the BMC Remedy SNMP Agent as an instruction to stop AR System. Changing the value of `arsState` from 2 to 1 will have the corresponding effect of instructing the BMC Remedy SNMP Agent to start the AR System.

The Remedy SNMP Agent can be used to 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 processes change their state (for example, if a process becomes inactive), the BMC Remedy SNMP Agent sends a trap (or a notification) to a trap receiver.

MIB-II

The 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 within these sections of MIB-II are not supported in the BMC Remedy SNMP Agent.

Remedy MIB

The Remedy MIB file, called `Remedy-ARS-MIB.txt`, defines all the objects managed by the 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 present in the `Remedy-ARS-MIB.txt` file are reserved for future use.

Sending traps

A trap is an asynchronous message that the 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 state. You can add a list of clients for receiving traps (also known as trap receivers) to the `arsnmpd.cfg` file.

The BMC Remedy SNMP Agent supports the following trap types:

- `coldstart`—Sent when the BMC Remedy SNMP Agent starts.
- `authentication failure`—Sent when a bad community string is supplied with an SNMP request. This type of trap is supported by SNMP versions 1 and 2c only, and must be enabled in the `arsnmpd.cfg` file.
- `arsStateChange`—Is a Remedy enterprise-specific trap type. The BMC Remedy SNMP Agent sends an `arsStateChange` trap when a change of state occurs for any of the following 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 that process
- The state of that 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 the value of `arsState`. When the process resumes, the trap contains a value of 1 for `arsState`.

Note: The 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 329.

SNMP configuration

Note: For information about configuring the BMC Remedy SNMP agent during installation, see the *Installing* guide.

The 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>.

The Remedy SNMP Agent uses the following configuration files:

Configuration file	Location	Purpose
Windows: arsnmpd.cfg	Windows: <ar_server_dir>\conf	Stores system information, access control information, and trap settings.
UNIX: arsnmpd.conf	UNIX: /usr/ar/<ar_system_name>/conf	
Windows: snmpd.conf	Windows: <ar_server_dir>\conf	Stores engine ID, number of Remedy SNMP Agent reboots, and SNMP v3 user account information.
UNIX: snmpd.conf	UNIX: /usr/ar/<ar_system_name>/conf	
Windows: armonitor.cfg	Windows: <ar_server_dir>\conf	Enables the Remedy SNMP Agent to monitor the AR System and to be started by armonitor.
UNIX: armonitor.conf	UNIX: /usr/ar/<ar_system_name>/conf	

The Remedy SNMP Agent uses the information in the arsnmpd and snmpd configuration files to initialize when the agent starts; therefore, you must restart the 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, the Remedy SNMP Agent will re-read the arsnmpd.cfg (arsnmpd.conf) file; so in this case, you do not need to restart the Remedy SNMP Agent.

The arsnmpd configuration file

Use the `arsnmpd.conf` (`arsnmpd.cfg`) file to configure any of the following information:

- System information (page 326)
- Access control information, which includes community strings and users (page 326)
- Trap configuration, which identifies the systems to which trap messages are sent (page 329)
- Location of the `armonitor` configuration file (page 330)

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> [<optional_argument>]  
# 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 within 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
<code>syslocation</code>	A string representing the location of the system running the Remedy SNMP Agent.
<code>syscontact</code>	A string representing contact information for AR System or the Remedy SNMP Agent, or both.

To define the system location, add the following directive:

```
syslocation <system_location>
```

To define the system contact, add the following directive:

```
syscontact <system_contact_information>
```

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 the Remedy SNMP Agent by querying the related MIB-II system group OIDs:

Directive	Description
<code>syslocation</code>	Used to populate sysLocation OID of MIB-II (1.3.6.1.2.1.1.6.0)
<code>syscontact</code>	Used to populate sysContact OID of MIB-II (1.3.6.1.2.1.1.4.0)

Access control information

For the Remedy SNMP Agent to respond to user requests, at least one directive specifying access control must be present in the `arsnmpd` configuration file. The 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 the 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 the 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 <community_string>
```

To establish a read-write community password, add the following directive:

```
rwcommunity <community_string>
```

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 the Remedy SNMP Agent, and it functions as a plain-text password (much like a community string). The 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. The 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. The 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 <user_name> [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 <user_name> [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, the Remedy SNMP Agent will default 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 331 for additional configuration requirements.

Trap configuration

Traps are unsolicited messages that the 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 Remedy SNMP Agent using an incorrect community string.

The 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 the 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 <system_name_or_IP_address> <community_string>
[<port_number>]
```

To send a trap formatted to the SNMP v2c standard, add the following directive:

```
trap2sink <system_name_or_IP_address> <community_string>
[<port_number>]
```

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 the 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`.

The 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 the Remedy SNMP Agent to send authentication failure traps. Setting the argument to 2 disables this feature.

Location of the armonitor configuration file

So that the Remedy SNMP Agent can interact with the AR System, uncomment the following line in the `arsnmpd.cfg` (`arsnmpd.conf`) file:

```
#arsmonitorfile <absolute_path_to_armonitor_file>
```

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 will be added to the snmpd file when the 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 328), you must include a corresponding createUser directive to this file as follows:

```
createUser <user_name> MD5 <authentication_password> DES  
<private_password>
```

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 (<user_name>).

Note: In SNMP v3, the authentication password that the user supplies to the 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 file

The armonitor configuration file permits the `armonitor` utility to start the Remedy SNMP Agent and to establish a link to it.

If you configured the Remedy SNMP Agent during installation, no modification to this file is necessary. If you did not configure the Remedy SNMP Agent during installation, you must edit the `armonitor.cfg` (`armonitor.conf`) file to enable `armonitor` to start and interact with the 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 will enable `armonitor` to start the 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 the Remedy SNMP Agent

You can start the Remedy SNMP Agent in any one of the following three ways:

- Using `armonitor`
 - If you configured SNMP during installation, `armonitor` will start the SNMP agent automatically after installation is complete.
 - If the SNMP agent process is terminated, `armonitor` will restart the SNMP agent.
- Using a command line with the following syntax:

```
arsnmpd -c <path to arsnmpd configuration file> udp:<port_number>
```

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 `arsystem` script (on UNIX).

Stopping the Remedy SNMP Agent

On Windows, the AR System is installed as a service. If you stop the AR System service, the Remedy SNMP Agent will also stop.

On UNIX, use the `arsystem` shell script to stop the AR System, which will stop all AR System processes including Remedy SNMP Agent.

If you use the Remedy SNMP Agent to stop AR System, the Remedy SNMP Agent will exist as an independent process that will not be under the control of the `armonitor`, the AR System service (Windows), or the `arsystem` shell script (UNIX). You must stop and restart the Remedy SNMP Agent manually using specific operating system methods (for example, by using Task Manager on Windows or using a `kill` command on UNIX).

If you want to stop the 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.) However, if the Remedy SNMP Agent is still a child process of `armonitor`, `armonitor` will attempt 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 have configured the Remedy SNMP Agent but it will not start.	The 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 <code>armonitor.cfg</code> (<code>armonitor.conf</code>) file and change the port number used by the Remedy SNMP Agent. Restart AR System.
You have instructed the Remedy SNMP Agent to change the value of <code>arsState</code> , but it will not respond to requests. All requests time out.	When the value of <code>arsState</code> is changed (to 1 or 2) by way of an SNMP set request, the Remedy SNMP Agent interprets this as a command to alter the state of the <code>arsSystem</code> to match the new value. It will not respond to additional SNMP requests until either of the following events have occurred: <ul style="list-style-type: none">■ The call to stop AR System returns.■ Eight minutes have elapsed (if attempting to set <code>arsState</code> to 1). Make sure that you have supplied a community string or a user name that has write permissions.
When you query for the current value of <code>arsState</code> , the result is 2 (down) even though the server is running.	Check the value of SNMP-agent-enabled in the <code>armonitor.cfg</code> (<code>armonitor.conf</code>) file. The value might be set to F. If so, set it to T and restart the AR System.

Problem or question	Resolution
You have set the value of <code>arsState</code> , but there is no change in the state of the AR System.	<p>Check the value of SNMP-agent-enabled in the <code>armonitor.cfg</code> (<code>armonitor.conf</code>) file.</p> <p>If this is set to F, you will not be able to stop or restart the AR System. Set this value to T and restart the AR System.</p> <p>If the value is already set to T, verify the following information:</p> <ul style="list-style-type: none">■ The <code>arsnmpd</code> process is running on the system with which you are trying to communicate.■ You have 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 <code>arsnmpd</code> process?	You do not need to restart the <code>arsnmpd</code> process. If you perform an SNMP set operation to change the value of <code>versionUpdateConfig.0 (.1.3.6.1.4.1.2021.100.11.0)</code> to 1, the Remedy SNMP Agent will re-read the <code>arsnmpd.cfg</code> (<code>arsnmpd.conf</code>) file.
You want to monitor IP traffic from the MIB-II group, but the Remedy SNMP Agent will not respond. Do all SNMP agents support MIB-II?	The 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 have configured your Remedy SNMP Agent to send trap messages to your Network Management Station (NMS), but there are no messages.	<p>Verify the following information:</p> <ul style="list-style-type: none">■ Check the arsnmpd.log file for trap-related error messages. If the Remedy SNMP Agent was unable to connect to a trap receiver, you will see the following entry: Error: Cannot create trapsink:<name of trap receiver>.■ 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 have supplied the correct community string for your NMS in the trapsink/trapsink directive present 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 have configured the Remedy SNMP Agent to send trap messages formatted according to v2 to an NMS that only supports v1, you will have to update your Remedy SNMP Agent configuration to send v1 traps instead.■ Any daemon processes required for receiving trap messages are running on your NMS.
The Remedy SNMP Agent is running, but you do not receive any information. All requests timeout.	<p>Verify the following information:</p> <ul style="list-style-type: none">■ You are using proper authentication. If you have provided the incorrect user name or password, your request will time out because the Remedy SNMP Agent does not respond.■ You have configured all access control components. (The Remedy SNMP Agent can run without an arsnmpd configuration file.) Check the arsnmpd.log file located in the db directory of your AR System installation. If you have not configured access control in the arsnmpd configuration file, you will see the following entry: Warning: no access control information configured. Remedy SNMP Agent cannot function in this state.

22 Using source control

Using a source control (SC) system helps you manage the development of your AR System applications by letting developers control access to specific system objects. Using an SC system can prevent developers from overwriting each other's work by enforcing check-in and check-out of objects. Source control is especially important when developers make mistakes and need to recover earlier versions of a definition, for example, using version history to recover deleted or modified system objects.

The following topics are provided:

- Integrating source control with AR System (page 338)
- Setting up source control with AR System (page 341)
- AR System source control options (page 347)
- Adding AR System objects to source control (page 349)
- Exporting AR System objects into source control (page 350)
- Importing definitions from source control (page 352)
- Removing objects in source control (page 355)
- Checking objects in and out of source control (page 356)
- Viewing history in source control (page 358)
- Getting the latest version of AR System objects (page 359)
- Displaying user information in source control (page 360)
- Running the source control client executable (page 361)

Integrating source control with AR System

This document assumes you are familiar with the details of operating your source control system, and understand how your source control database operates. BMC Remedy Administrator will *not* prevent you from improperly setting up your source control environment; for example, having the AR System server pointing to different source control projects and different administrators overwriting each other's changes. If you have questions about implementation, see the documentation provided with the source control software.

When you integrate source control with AR System, application developers can quickly see in BMC Remedy Administrator if an object is checked out of the source control database and who its current owner is, as shown in the following figure.

Figure 22-1: Source control in AR system

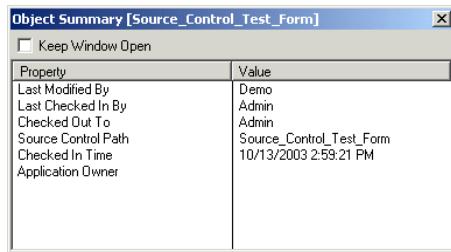
The screenshot shows the 'Server Window 2 (SHOFM21)' interface. The left pane is a tree view of the server structure under 'Servers'. The 'Forms' node is expanded, showing various form types and their properties. The right pane is a table titled 'Forms:' with the following data:

Name	Type	Modified	Web Alias	Lock State	Check Out User
Distributed Pool	Regular	10/10/2...	None	Not in SC;	
FB:Alarm Events	Regular	10/13/2...	None	Not in SC;	
FB:Alarm Monitor	Regular	10/13/2...	None	Not in SC;	
FB:CumulativeServerStatistics	Regular	10/13/2...	FBCumul...	None	Not in SC;
FB:DataSource	Regular	10/13/2...	None	Not in SC;	
FB:DataSourceVariables	Regular	10/13/2...	None	Not in SC;	
FB:Flashboards	Regular	10/13/2...	None	Not in SC;	
FB:History	Regular	10/13/2...	None	Not in SC;	
FB:History Summary	Regular	10/13/2...	None	Not in SC;	
FB:NonCumulativeServerStatistics	Regular	10/13/2...	FBNonC...	None	Not in SC;
FB:Variable	Regular	10/13/2...	None	Not in SC;	
FB:Variable Attributes	Regular	10/13/2...	None	Not in SC;	
Flashboard Server Statistics Sample	Display...	10/13/2...	Flashboa...	None	Not in SC;
Group	Regular	10/10/2...	Group	None	Not in SC;
Home Page	Display...	10/10/2...	HomePage	None	Not in SC;
inetorgperson	Vendor	10/10/2...	None	Not in SC;	
JOHN_FORM	Regular	10/10/2...	JOHNFO...	None	Not in SC;
Report	Regular	10/10/2...	None	Not in SC;	
ReportCreator	Vendor	10/10/2...	None	Not in SC;	
ReportSelection	Regular	10/10/2...	None	Not in SC;	
ReportType	Regular	10/10/2...	None	Not in SC;	
Sample:Cities	Regular	10/10/2...	SampleCi...	None	Not in SC;
Sample:ClassCentral	Display...	10/10/2...	ClassCen...	None	Not in SC;
Sample:Classes	Regular	10/10/2...	SampleCl...	None	Not in SC;
Sample:DialogYesNo	Display...	10/10/2...	SampleY...	None	Not in SC;
Sample:Enrollments	Regular	10/10/2...	SampleE...	None	Not in SC;
Sample>ShowWeather	Display...	10/10/2...	SampleS...	None	Not in SC;
Server Events	Regular	10/10/2...	None	Not in SC;	
Server Statistics	Regular	10/10/2...	None	Not in SC;	
Source Control Test Form	Regular	10/13/2...	None	In SC: Admin	
User	Regular	10/10/2...	User	None	Not in SC;

Source control information includes if the object is in source control, whether it is checked out, the last check-in time, and developer comments, if any. If developers do not have access to the AR System server or do not have check-in and check-out privileges in source control, they cannot make changes to the object.

To view detailed source control information about a server object, right-click on the object, and choose Object Summary to view the Object Summary dialog box.

Figure 22-2: Object summary dialog box



Note: You cannot place groups and distributed mappings or pools under source control because they are actually entries inside special forms. Additionally, you cannot check extension objects (such as Flashboards objects) into source control.

Enforced and advisory modes

When you integrate your source control software with AR System, you have the option of defining whether the AR System server is in “enforced” or “advisory” mode.

If you set the server to **Enforced mode**, BMC Remedy Administrator:

- Maintains consistency between the AR System server and the source control database.
- Prevents accidental changes from taking place.

By contrast, in **Advisory mode**, AR System does not enforce the links between the AR System server and the source control database. You run the risk of allowing discrepancies to creep in between the AR System server and the source control database. If a discrepancy exists between the server and the database, *AR System will always assume the AR System server is correct.*

In Advisory mode, the AR System server and the source control database can easily get out of synchronization. For example, administrators might:

- Create API programs that modify objects (for example, disabling an active link through the driver program).
- Allow multiple developers in AR System.

To update definitions in source control, you can check out every object you want to update and check in the latest changes. To return to a previous version of an object stored in source control, get the version you want from source control, and then import the definition file using the Import in Place option. For information, see “Importing definitions from source control” on page 352.

Table 22-1 explains the differences between Enforced and Advisory modes when creating, opening, or modifying server objects.

Table 22-1: Differences between enforced and advisory modes

	Enforced mode	Advisory mode
Creating Objects	By default, automatically adds server objects to source control.	Can create objects without checking them into source control. Option provided in Save dialog box for adding object to source control.
Opening Objects	Cannot open objects not in the source control database. Warning appears that objects do not exist in source control.	Can open objects without checking them out from source control first.
Modifying Objects	Must check out objects from source control <i>first</i> before modifying them. Otherwise, warning appears. If you make any changes to the object and want to save them, you can still check out the object, and save it. However, if you close the object without checking it out, no changes are saved.	Can modify objects without checking them out from source control first.

For information about configuring an AR System server for source control, see the *Configuring* guide.

Setting up source control with AR System

The following procedure provides an overview of how to use source control with AR System.

Important: The following procedure is provided with the understanding that, while the procedure is valid, BMC Remedy does not officially support integration with any source control system. Please see the compatibility matrix for more information. The compatibility matrix always provides the latest, most complete information about what is officially supported.

► To set up source control with AR System

- 1 Install the source control client and server software *before* integrating it with BMC Remedy Administrator.

Make sure that AR System administrators and AR System application developers are properly licensed for the source control system you are using. In addition, make sure the project name is a local directory on their computers. Source control administrators can set properties like login name and password as well as specific properties from the source control provider. The login name used by the administrator on the source control system can be different from that on AR System.

For integration with PVCS, see “Integrating AR System source control with PVCS” on page 344. For integration with ClearCase, see “Integrating AR system source control with ClearCase” on page 345. Also see “To display user information in source control” on page 360.

- 2 Configure AR System with the source control database.

In BMC Remedy Administrator, open the Server Information dialog box for the server, and click the Source Control tab.

You must choose a source control provider, the level of integration with AR System (Enforced or Advisory), the fully qualified target directory (the Project Name) to the source control database, and whether check-in and check-out comments are required.

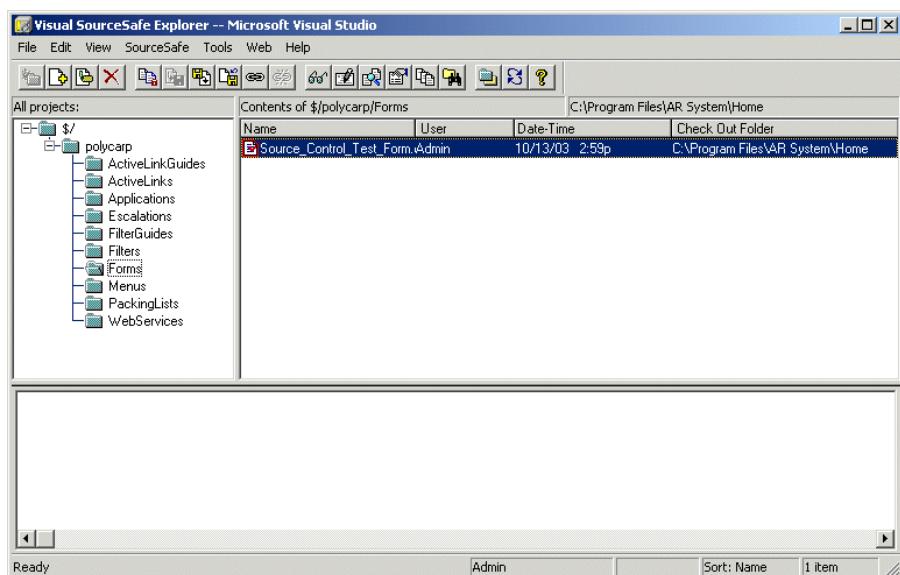
WARNING: Do not use source control with a mixed AR System environment. For example, if you implement source control with AR System, make sure all developers are using the same versions of AR System.

When accessing the source control database, you will need the login name and password of the source control client that is used to connect to the source control database.

3 Create a source control project.

To create a project, in the Source Control tab in the Server Information dialog box, click the Browse button and select a project. The source control database is a file-based system of .def files. A collection of directories to store .def object files (for example, forms, filters, and so on) will be created in the source control database, as shown in Figure 22-3.

Figure 22-3: Source control database (Form definition files displayed)



Note: These .def files are text files, not binary files. They are the same type of files used for importing and exporting definitions of AR System server objects.

The read-only Project Name field on the Source Control tab shows which source control database you have selected.

You can use the same settings for all servers that you want to connect to. You also have the option to enable or disable source control for all servers by default.

- 4 After this configuration is complete, you are prompted to log in again to the AR System server for your changes to occur.

Depending on your source control configuration, when you log in to BMC Remedy Administrator and select a server, you might be prompted to log in to your source control client as well.

After login is complete, you see that the source control toolbar buttons and menus in BMC Remedy Administrator are enabled.

- 5 Set the user information so that you can properly check out and check in files.
 - a In the Server Window of BMC Remedy Administrator, select the appropriate server.
 - b Choose Tools > Source Control > User Info.
The User Info dialog box appears.
 - c Enter the user name and password.
 - d Click OK.

Important: Before you check objects in or out, you must log in to the source control system.

- 6 Add AR System objects to the source control database, except for distributed mappings and groups.

See “Adding AR System objects to source control” on page 349 and “To export object definitions to source control” on page 350 for more information.

- 7 In BMC Remedy Administrator, create, open, or modify server objects as normal.

For differences between enforced and advisory modes when creating, opening, or modifying objects, see Table 22-1 on page 340.

- 8 Save your changes to the objects.

The object is now saved to the AR System server. However, changes to objects are *not* updated in source control until you check them in. For more information about checking objects into source control, see “To check in AR System objects to source control” on page 357.

- 9 Check the objects into source control by selecting them and clicking the Check In button on the source control toolbar in BMC Remedy Administrator.

In Advisory mode, if the object had not been previously checked into the source control, BMC Remedy Administrator will automatically create a version. You can also check an object back into the source control database without saving it first, but your changes will not be included in the .def file.

For more information, see:

- “To remove AR System objects from source control” on page 355
- “To undo checkout of AR System objects in source control” on page 357
- “To show the history of AR System objects in source control” on page 358
- “Refreshing the status history in source control” on page 359
- “To run the source control client executable” on page 361 to learn about starting the source control client from BMC Remedy Administrator.

Integrating AR System source control with PVCS

For PVCS to work correctly with AR System, you must perform the following procedure.

Important: The following procedure is provided with the understanding that, while the procedure is valid, BMC Remedy does not officially support integration with any source control system. Please see the compatibility matrix for more information. The compatibility matrix always provides the latest, most complete information about what is officially supported.

► To integrate AR System source control with PVCS

- 1 Add a valid user name to the User Info option in the Source Control menu from BMC Remedy Administrator when you right-click the object.
- 2 From BMC Remedy Administrator, select the main Project Folder to use for source control, and not a sub-folder.

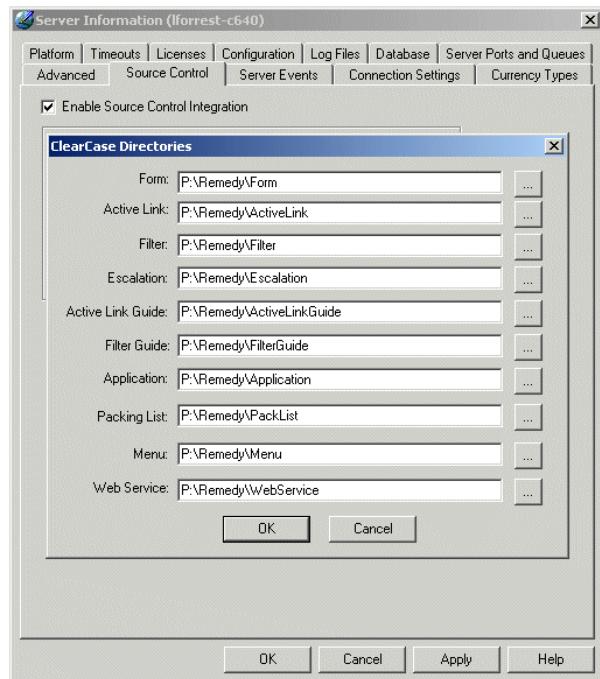
Integrating AR system source control with ClearCase

For ClearCase to work correctly with the AR System, you must perform the following procedure.

Important: The following procedure is provided with the understanding that, while the procedure is valid, BMC Remedy does not officially support integration with any source control system. Please see the compatibility matrix for more information. The compatibility matrix always provides the latest, most complete information about what is officially supported.

► To integrate AR System source control with ClearCase

- 1 After installing ClearCase on your system, create and mount your ClearCase VOB (versioned object base), which is the repository on the server's disk.
- 2 Create folders for forms and workflow objects under your ClearCase VOB.
- 3 In the Source Control tab of the Server Information window, add ClearCase as the Provider Name.
- 4 In the Project Name field, click Browse and add objects to your source control project.

Figure 22-4: ClearCase directories

You can edit the directories used to store AR System objects where needed. Your changes will be saved to the ar.conf (ar.cfg) file, as follows:

```
SCC-Target-Dir:  
1~P:\Remedy\Form~2~P:\Remedy\ActiveLink~3~P:\Remedy\Filter~4~P:\Remedy\Escalation~5~P:\Remedy\ActiveLinkGuide~6~P:\Remedy\FilterGuide~7~P:\Remedy\Application~8~P:\Remedy\PackList~9~P:\Remedy\Menu~10~P:\Remedy\WebService
```

The ar.conf (ar.cfg) file limits the list of AR System objects that you can include under source control to 255 characters. If you use more than 255, your integration will not work because AR System will not have enough information.

For more information about the ar.conf (ar.cfg) file, see the *Configuring* guide.

AR System source control options

The following toolbar buttons and menu items are available in BMC Remedy Administrator for using source control integration with AR System. You can also access these functions through the Tools menu or by right-clicking an object in the Server Window.

Toolbar item	Button/Menu	Description	Page
	Get Latest Version	<p>Performs one of the following tasks:</p> <ul style="list-style-type: none"> ■ Writes a version of the most recent AR System definition file (.def) from source control into a directory you specify. ■ Imports in place the most recent version of the object into the AR System server. <p>When you “get” an object, you are merely retrieving a copy in its latest revised state. The file is not locked from other system administrators, who in the meantime can check out and lock the file for their own use.</p> <p>To “get” an earlier version of an object, use the Show History function, or use the source control client itself.</p>	page 359
	Check In	Updates source control with the latest version of the AR System objects. When you check in an object, the file is no longer locked, and other system administrators can use it or modify it.	page 357
	Check Out	Copies the latest version of one or more selected AR System objects from the current project into your current working folder. When you check out an object, the file is locked, and no other system administrator can modify it.	page 356
	Undo Checkout	Cancels the Check Out operation and undoes all changes. You must have a working folder set in your source control client for this command to work properly. The file is no longer locked by you, and other system administrators can check out the file for their own use. The previous version of the file is retained in the source control project.	page 357
	Add To Source Control	Adds an AR System object (.def file) to the source control database. The file is archived in the source control project.	page 349

Toolbar item	Button/Menu	Description	Page
	Remove From Source Control	Removes .def object files from the source control database. Removing files from source control does not remove them from the AR System server.	page 355
	Show History	<p>Displays a list of the past versions of the AR System objects in your source control project. History reports summarize information about revisions of the object.</p> <p>In addition, showing the history of a file gives you the option of “getting” a copy of the object, in case you made a mistake and need to revert to an earlier version. Depending on your source control client, you also can compare the differences with the current and previous files (you can perform a “diff”).</p>	page 358
	Show Differences	Reserved for future use by AR System.	
	User Info	Displays login information about the source control user.	page 360
	Refresh Status	Refreshes in BMC Remedy Administrator the current information about AR System objects from the source control database.	page 359
	Run Source Control Client	Starts source control client executable to its project directory.	page 361

You can hide or view the source control toolbar by choosing View > Toolbars > Source Control.

Adding AR System objects to source control

When you enable source control integration with AR System, all subsequent objects that you create will be added to source control if the system is in Enforced mode. However, server objects on existing systems or objects in Advisory mode will not exist under source control *until* you add them. Use the following procedure to add multiple objects of the same type (for example, forms) into source control.

To add a large number of objects of different types (for example, forms, active links, and so on) at the same time into the source control database, use the Export Definitions to Source Control feature.

► To add AR System objects to source control

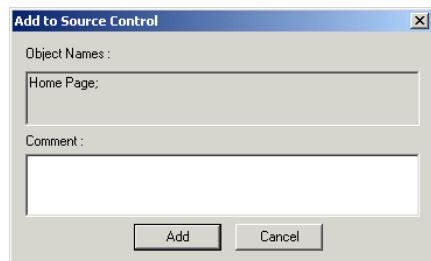
- 1 In the server window, select one or more AR System objects to add to source control.

You can use the SHIFT and CTRL keys to select multiple objects.

- 2 Choose Tools > Source Control > Add to Source Control.

The Add to Source Control dialog box appears.

Figure 22-5: Add to source control dialog box



The server window shows that the object has been added to source control.

- 3 Enter any appropriate comments.
- 4 Click Add.

The server window is refreshed to show that the server object has been added to the source control database. If you open the source control client, you will also see the object's .def file added to the source control database.

Exporting AR System objects into source control

You can export a file into source control. The export feature is a valuable time-saver because you can add multiple forms, filters, menus, and so on into source control as needed, instead of having to add objects into source control by individual type. This option allows you to update the current version of your project in source control by taking a “snapshot” of the AR System server, which you can then export into your source control system. The export feature could function for you as a system backup tool to perform exports of the AR System server, as needed (for example, weekly or monthly).

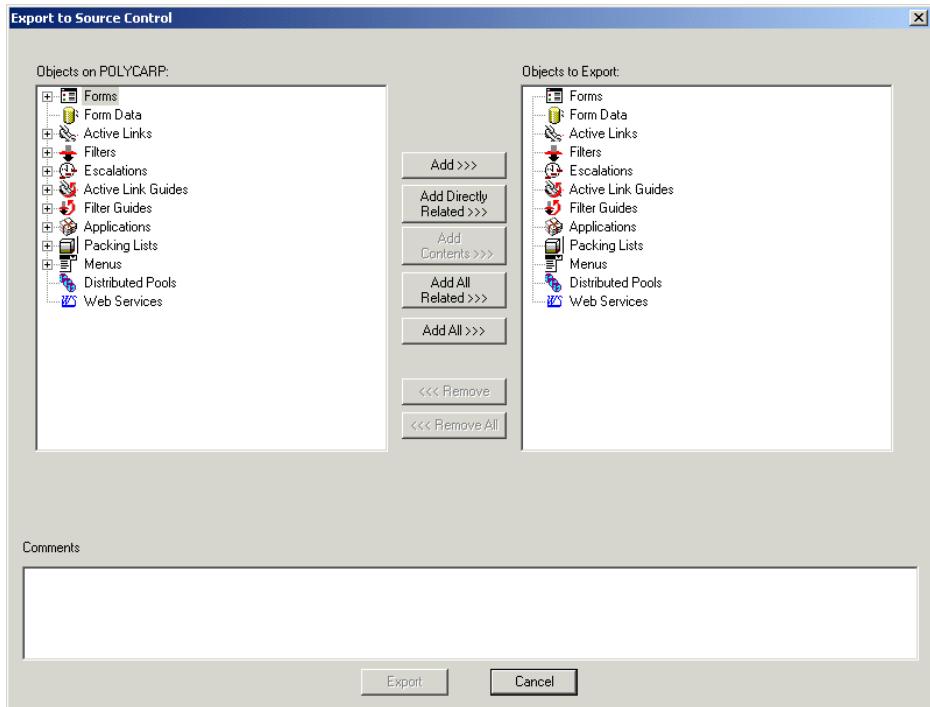
► To export object definitions to source control

- 1 Select a server to administer.
- 2 Choose Tools > Export Definitions > To Source Control to export definitions into source control.

Use the Export to Source Control dialog box (Figure 22-6 on page 351) to select the definitions that you want to export into source control. Definitions for each object type appear where at least one form, menu, filter, escalation, active link, application, guide, and packing list has already been created. If you see a plus (+) next to an object type, at least one object of that type exists.

All object definitions from the specified server are displayed in the *<Objects>* on *<server>* list. To see a more detailed list of available objects, click the plus sign (+) next to each object icon.

Figure 22-6: Export to source control dialog box



- 3** Move the appropriate objects from the Objects on <server> list to the Objects to Export list.

For a complete explanation of the buttons on this dialog box, see the *Form and Application Objects* guide.

- 4** Enter appropriate version information in the Comments box when exporting definitions into source control.

Depending on how you configured your source control system, comments are mandatory or optional. For information about Enforced and Advisory modes and configuring an AR System server for source control, see “Enforced and advisory modes” on page 339 as well as the *Configuring* guide.

- 5** Click Export.

Status messages for each server object definition appear as the objects are added to source control.

- 6** When the export to source control is finished, click Cancel to close the Export to Source Control dialog box.

Importing definitions from source control

Importing definitions from source control is especially important for keeping the AR System server and your source control system synchronized. For example, an AR System server might contain a version earlier than the one existing in the source control database, because source control might have been updated due to changes on the same object from another AR System server.

The options for importing definitions from source control are:

- **To a File**—Copies definitions from your source control system into a .def file. This option is useful for importing new definitions (for example, from a different system) into your AR System server. First, you would import the definitions from source control to create a .def file, and then use the Import Definitions From Definition File feature to copy the definitions into the AR System server.
- **Import in Place**—Imports definitions from source control and overwrites existing objects on the server. This option is useful for restoring object definitions, for example, if the AR System server became corrupt and you wanted to restore them from source control.

Import in Place uses the following process:

- Checks if the object is available to check out, imports the object in place, and checks the object back in to source control.
- If the object is already checked out by you, the process imports the object in place, and leaves the object checked out.
- If the object is checked out to someone else, you see an error message for that object, and the process continues for other objects.

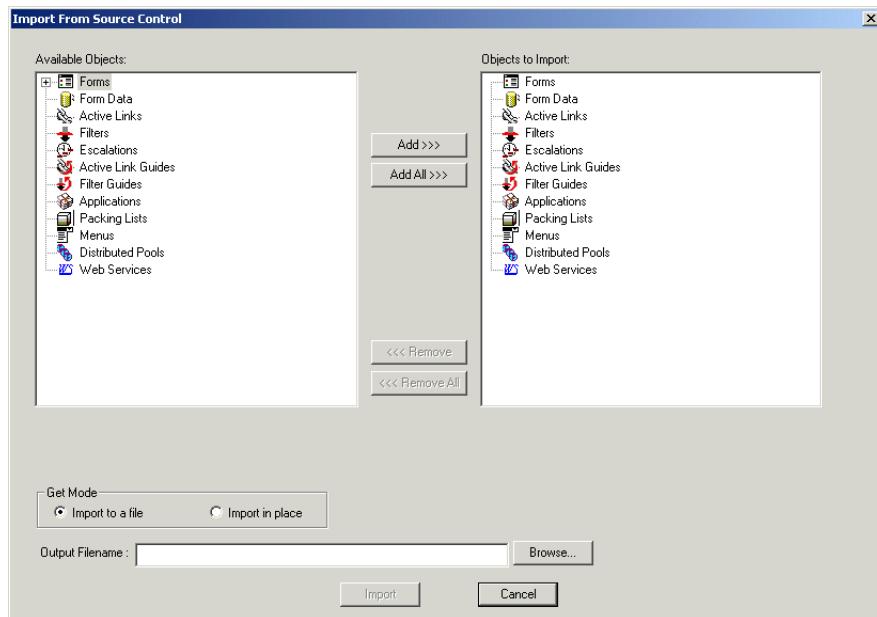
WARNING: If you use the Import in Place option to import a form definition, back up the underlying form data first from the old form. The Import in Place operation first deletes the form definition and the data, and then it recreates the form on the server. “Importing in place” might also affect workflow; for example, if active links see fields that no longer exist.

► To import definitions from source control

- 1 Select a server to administer.
- 2 Choose Tools > Import Definitions > From Source Control.

The Import From Source Control dialog box appears.

Figure 22-7: Import from source control dialog box



Use this dialog box to select the definitions that you want to import into your source control system. Definitions for each object type appear where at least one form, menu, filter, escalation, active link, application, guide, or packing list has already been created. If you see a plus sign (+) next to an object type, this means that at least one object of that type exists.

- 3 Click the appropriate object or objects.

All object definitions are displayed in the Available Objects list. To see a more detailed list of available objects, click each object icon.

4 Move the objects from the Available Objects list to the Objects to Import list.

You cannot have two objects with the same name on a single server. If an object in the import file has the same name as an object on the destination server, take one of the following actions before completing the import:

- Remove the object with the same name from the destination server.
- Rename the object with the same name on the destination server.

To rename an object, select it from the Objects to Import list, click again to highlight the name, and type a new name.

WARNING: If you rename a form in the Import From Source Control dialog box and then import the newly named form to the destination server, the form will not retain the connection it had with its associated workflow on the source server. To maintain the connection between a renamed form and its associated workflow, you must rename the form in BMC Remedy Administrator before you create the object definition file. For more information about renaming forms, see the *Form and Application Objects* guide. If you rename a menu, active link, filter, or escalation in the Import From Source Control dialog box, you must reattach it to the appropriate fields on the destination server.

5 In the Get Mode region, select one of the following options:

- **Import to a File**—Copies the object definitions to a .def or .xml file.
- **Import in Place**—Overwrites an existing object with the version from source control.

6 If you select the Import to a File option, enter a path and .def file name in the Output field.

You can click Browse to locate a specific .def file, which will be overwritten with the new information.

7 Click Import to import the definitions to the destination server.

The new definition names are added to the corresponding categories of the server window. When the import is finished, an Import Complete message appears.

8 Click OK to acknowledge the message.

9 Click Cancel to close the Import Definitions dialog box.

Removing objects in source control

Removing or deleting an object from source control is not the same as deleting an object from the AR System server. Conversely, deleting an object from the server does not delete it in the source control database. This is because the object in source control might be used by other AR System servers accessing the same source control database. To delete the object totally, you must remove it from the AR System server *and* from the source control database.

If you delete the object from the AR System server but not from source control, BMC Remedy Administrator will add a comment in source control, stating that the object was deleted in AR System and will specify the fully qualified path name of the server where the operation was completed.

If you remove an object referenced in a container from source control, for example, an active link that is used in a guide, the server automatically removes the reference from the container but does not remove the container.

When you remove an object from source control, you lose the ability to track the history of any changes made to the object.

Note: Removing an object from source control works differently in Enforced mode than in Advisory mode. If you remove an object from source control in Enforced mode, the system still *strictly* maintains source control over all AR System objects and will not let you modify them even if they do not exist in source control. However, Enforced mode will let you delete objects from the AR System server after they have been removed from source control.

► To remove AR System objects from source control

- 1 In the Server Window, select one or more AR System objects to be removed from source control.
- 2 Choose Tools > Source Control > Remove from Source Control.

A warning message appears, informing you that when removing the object from the source control database, you will also lose history data.

- 3 Click OK.

The Server Window refreshes to display that the server object has been removed from source control.

Checking objects in and out of source control

The following procedures describe how you check objects in to and out of source control through BMC Remedy Administrator.

Checking out AR System objects

Use the following steps to check out and lock a file exclusively for your use. With Enforced mode, other administrators will not be able to modify and save changes on an object that you have checked out.

► To check out AR System objects from source control

- 1 In the server window, select one or more AR System objects to check out from source control.

You can use the SHIFT and CTRL keys to select multiple objects.

- 2 Choose Tools > Source Control > Check Out.

The Check Out from Source Control dialog box appears.

Figure 22-8: Check out from source control dialog box



The server window lists the user who has checked out the object.

- 3 Enter any appropriate comments.
- 4 Select Check Out related objects to include related AR System objects.

This feature is important, for example, if you are modifying an active link that is attached to a form or a guide. This is because changing workflow in one place has a “ripple effect” on other related server objects. Simply changing the name of an active link breaks the workflow with related objects like forms or guides. Checking out related objects is a useful feature in avoiding potential conflicts with other system administrators.

- 5 Click OK.

The server window shows what objects are checked out by each user.

Undoing a check-out of AR System objects

Cancelling the Check Out operation leaves the object still checked into the source control system. In Enforced mode, this means you cannot modify the object. However, if you undo a check-out, any changes saved to the AR System server since the last check-in will *not* be undone. Undoing a check-out will only clear the Checked Out To object property.

► To undo checkout of AR System objects in source control

- 1 In the server window, select one or more AR System objects that are checked out from source control.
- 2 Choose Tools > Source Control > Undo Check Out.
A warning appears.
- 3 Click Yes (or click Yes All).

The server window shows that the server object has not been checked out to you.

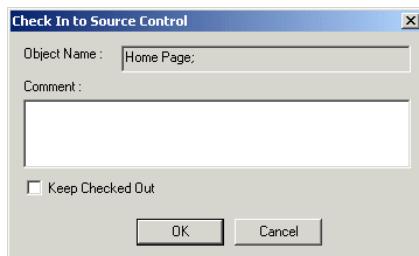
Checking in AR System objects

After you have modified an object and saved it, use the following procedure to check it back in to the source control database.

► To check in AR System objects to source control

- 1 In the server window, select one or more AR System objects to check in to source control.
- 2 Choose Tools > Source Control > Check In.

The Check In to Source Control dialog box appears.

Figure 22-9: Check in to source control dialog box

- 3 Enter any appropriate comments.
- 4 Click Keep Checked Out to update the object in the source control system but keep it checked out.

In Enforced mode, the system will check the object into the source control database, and then immediately check it back out, *as one operation*. This feature is useful because the object is still locked for you, and no other administrators will be able to modify it.

- 5 Click OK.

The server window shows that the server object has been checked in to the source control database—that is, unless you kept the object checked out.

Viewing history in source control

Administrators can also look at the history of a given object in source control, update the source control status on the server window, or undo a check-out from source control. When you undo a check-out, any changes saved to the AR System server since the last check-in will not be undone.

Use the following procedure to view the history of objects (the comments system administrators add over the history of the object) in the source control database.

► To show the history of AR System objects in source control

- 1 In the server window, select one or more AR System objects you want to check.
- 2 Choose Tools > Source Control > Show History.
The History Options dialog box for your source control software appears.
- 3 Define the history information, as appropriate.

Refreshing the status history in source control

Refreshing the status list in BMC Remedy Administrator is especially useful for ensuring that:

- Your .def files were added to the source control database.
- You have the latest version of the .def files.
- Another developer does not have the file checked out.

► To refresh the status in the file list

- 1 In the server window, select one or more AR System objects for the status you want to refresh.
- 2 Choose Tools > Source Control > Refresh Status.

The information in the AR System server will be refreshed from the source control database.

Getting the latest version of AR System objects

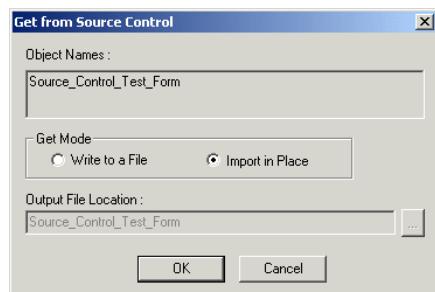
Getting the latest version of an object copies the most recent version of the object from the source control database into the AR System server, or creates a copy in a location you specify.

► To get the latest version of AR System objects from source control

- 1 In the server window, select one or more AR System objects whose latest version you want to get from source control.
- 2 Choose Tools > Source Control > Get Latest Version.

The Get from Source Control dialog box appears.

Figure 22-10: Get from source control dialog box



3 Perform one of the following steps:

- To copy the object definition into a .def file, select Write to a File, and click the Browse button (...) to select a file location.
- To copy (and overwrite) a server object from source control into the AR System server, select Import In Place (the default). To copy a large number of objects from source control into the AR System server, see “Importing definitions from source control” on page 352.

4 Click OK.

The object’s .def file is copied into your working directory.

Displaying user information in source control

Use the following procedure to display user information and source control options about the user.

► **To display user information in source control**

- 1** In the Server Window, select a server for the user information you want to display.
- 2** Choose Tools > Source Control > User Info.
The User Info dialog box appears.
- 3** Enter the user name and password.
- 4** Click Advanced Properties to define general options for the source control project.

The source control options that appear display the properties you can set for the selected project.

- 5** Click OK.

Note: To avoid security issues if multiple users are using the same SourceSafe and BMC Remedy Administrator clients, or if there is a problem logging in to SourceSafe, remove the SSDIR and SSUSER environmental variable settings from the Environment tab in the System Properties dialog box in the MS Windows Control Panel.

Running the source control client executable

You can launch the source control client from BMC Remedy Administrator. This feature might be necessary and useful in some source control environments that support multiple check-outs and merges.

► To run the source control client executable

- 1 Choose Tools > Source Control > Run Source Control Client to open the source control client for the project.
- 2 Return to BMC Remedy Administrator, as needed.

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 364)
- Application licensing overview (page 365)
- Configuring your applications to make them licensable (page 367)
- Applying the application license on your server (page 369)
- Assigning application licenses to users (page 371)

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 have created BMC Remedy applications that you want to license for sale, see the Remedy alliance website (<http://supportweb.remedy.com/appreg>) 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 will receive a licensing error when they try to perform any data-related operations on forms in the unlicensed application. In other words, they will not be able to 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. There are three types of application user licenses:
 - Fixed
 - Floating
 - Read licenses

Fixed and floating licenses have no restrictions and allow users to get, modify, search, create, delete, and merge entries. Read licenses allow 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	<ul style="list-style-type: none">■ If a merge operation results in a <i>create</i>, does not need application or application user license.■ If a merge operation results in a <i>modify</i>, 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 will automatically use a read license by default. Guest users will 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, has created a new application, MusicManager, that they want to license for sale. The following steps describe the process to license their new application. Typically, ISVs will perform step 1 through step 4. Customers who have purchased the application will perform step 5 and step 6.

- 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, the following naming convention is used:

<vendor_name>:<application_name>

In this example, the name is XYZ:MusicManager.

In most cases, you will not need to worry about conflicts with naming conventions for your application. Even if XYZ has a competitor with their own MusicManager application, the vendor name prefix will guarantee uniqueness in nearly all cases.

Registration also includes providing application-specific information to BMC, so that the license can be appropriately generated.

If you have questions, go to the Customer Support website (<http://supportweb.remedy.com/appreg>) and verify that the application name is unique or has not already 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 364:

- Customers must obtain application licenses so that they can access the form data in the application.
- Customers must obtain user-fixed or user-floating licenses based on individual forms to access form data.

In this example, XYZ Corporation decides to make MusicManager application licensable *and* user licensable. The MusicManager application includes just three forms: MusicManager Configuration, MusicManager Songs, and MusicManager Singers.

If the customer tries to access data in any of these forms, they will receive an error if they have not applied for a valid MusicManager application license.

Further, XYZ Corporation decides to designate *only* the MusicManager Songs form to be user-licensable (shown in Figure 23-1 on page 368). If any users attempt to submit or modify data in this form, they will receive an error if they do not have a MusicManager user license (fixed or floating). Since the ISV has chosen not to make the MusicManager Singers form user-licensable, any user can create or modify MusicManager Singers without a MusicManager user license.

For detailed steps, see “Configuring your applications to make them licensable” on page 367.

Step 4 BMC generates the license keys for the registered licensable application.

After you register your application, the application vendor obtains licenses through the Customer Support website. The process for obtaining an application license is similar to obtaining a server license. Application licenses are tied to a specific Host ID. Licenses can have an expiration date.

In the example, after selling the MusicManager application to a new customer, the sales representative at XYZ Corporation obtains the Host ID from the system where the application is installed, logs in to the Customer Support website, and provides this and other information necessary to generate the license keys. Since the XYZ:MusicManager application has been previously registered, the licenses appear on the website, similar to obtaining a server license.

-
- Step 5** (Customers only) In BMC Remedy Administrator, choose File > Manage License to apply the license on your AR System server.

For information, see “Applying the application license on your server” on page 369.

- Step 6** (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).

Here you would open the User form in BMC Remedy User 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 normal manner. For information, see “Assigning application licenses to users” on page 371.

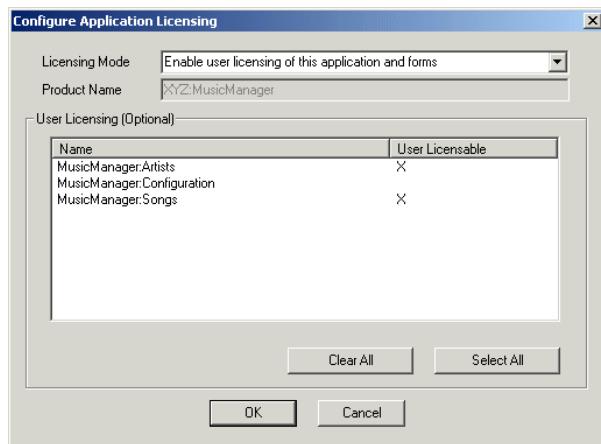
Configuring your applications to make them licensable

ISVs use the following procedure to configure your applications to make them 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 has been made licensable, the process cannot be reversed, even by the ISV. In addition, after an application is user licensed and the forms are checked, those forms cannot be unchecked and you cannot revert to application licensing or non-licensing 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 Create an application.
- 2 Choose Application > License Application.

Figure 23-1: Configuring your application license

- 3 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 installed 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.

- 4 In the Product Name field, enter the application license string, for example, XYZ:MusicManager.

- 5 If you choose to enable user licensing of this application and its forms, perform the following actions:
 - Click inside the User Licensable list for the form that you want to license. An “X” appears next to forms that you select.
 - Click Select All to license *all* the forms in your application.
 - Click Clear All to clear all the forms from the User Licensable list.
- This step is optional for applications, and is needed only if you want to make forms in your application user-licensable.
- 6 Click OK to finish making your applications licensable.

Applying the application license on your server

After customers have obtained the application license key from BMC, they will use the following procedure to apply it to their server. Customers license their new application, MusicManager, just like any other AR System application. The only difference is the unique naming convention of the application itself, indicating the application comes from an ISV.

Note: Customers use the same procedure to activate fixed and floating user licenses.

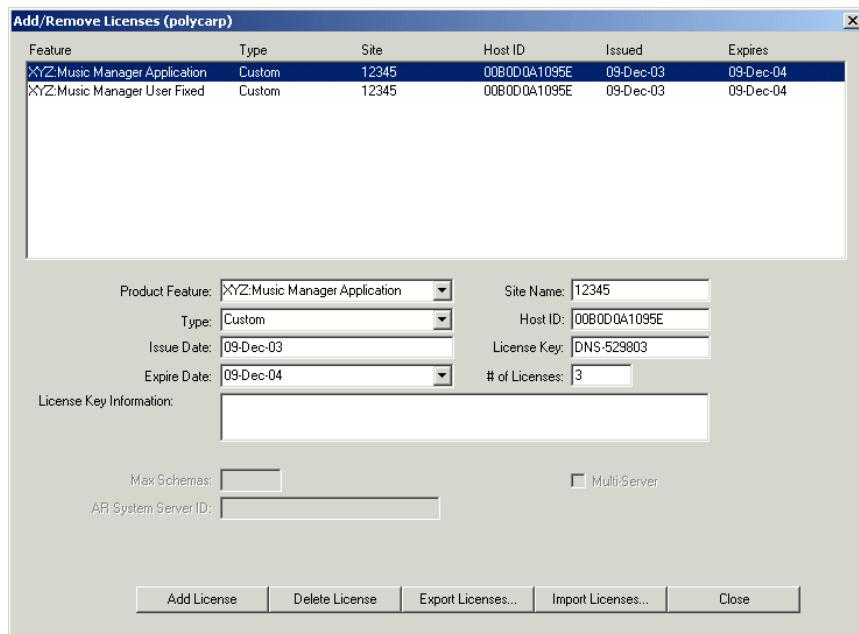
► To apply the application license key to your server

- 1 Log in to BMC Remedy Administrator with administrator permissions.
- 2 Choose File > Licenses > Add/Remove Licenses.

The Add/Remove Licenses window appears.

- 3 After receiving the application license key, enter the application license information into the Product Feature and Type fields.

Figure 23-2: Applying application license information into Add/Remove licenses window



Here customers must type the application license information into the Product Feature and Type fields. This example shows how customers would license the MusicManager application for 3 User Fixed licenses.

When you apply the license to your application, you must use the format <*product_name*> Application, for example, XYZ:MusicManager Application.

- 4 In the Site Name field, enter your customer support ID, including any leading zeros.
- 5 Enter the remaining information *exactly* as specified (including capitalization) in the appropriate fields.

All fields, including the appropriate license key for the Product Feature and the issue and expiration date (if any), *must* match the information that Customer Support provides.

Like any other AR System licenses, licenses have an expiration dates. Application licenses must be renewed when they expire. When these licenses are renewed depends on the customer's support contracts and agreements.

6 Click Add License.

Customers are prompted that they have received a non-exclusive, non-transferable license. The application is now licensed on the server.

7 Reboot your server for the licenses to take effect.

Assigning application licenses to users

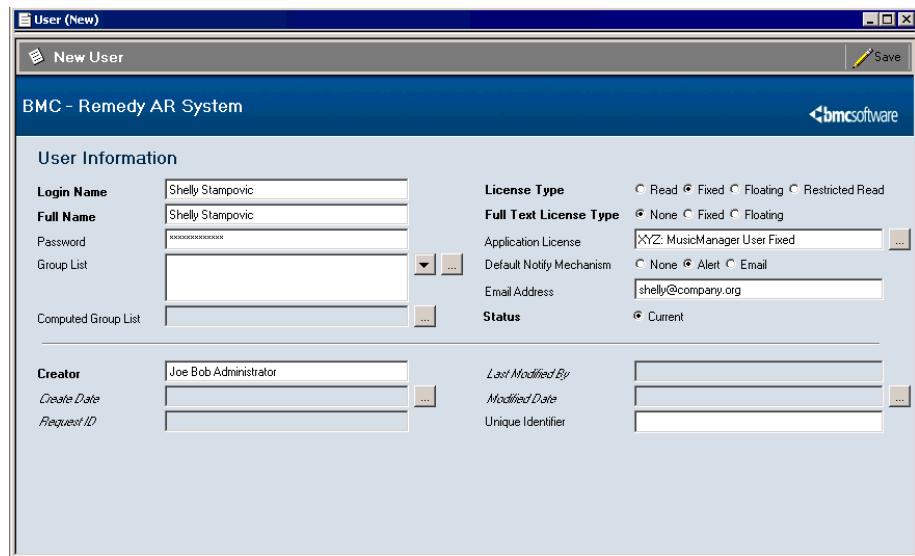
After customers have installed the application user license keys for the application from BMC Remedy, they use the following procedure to apply it to their user community.

Note: These application licenses are in addition to AR System licenses. This method of licensing is completely separate from the need for AR System User licenses to modify entries.

► To assign licenses to users

- 1** Log in to BMC Remedy User with administrator permissions.
- 2** Open the User form in new mode to create a new user. Otherwise, open the User form in search mode and perform a query to retrieve a list of currently defined users.
- 3** Enter information in the appropriate fields for creating new users or modifying existing users.

Figure 23-3: Entering information into the User form



- In the Application License field, enter the name of the application and the appropriate license type, as shown in Figure 23-3.

Note: Use the Application Licenses field to give users their fixed and floating licenses to your application. Do *not* specify user license information with the License Type field.

Use the following syntax when providing users with application licenses:

```
<vendor_name>:<application_name> <user> <type_of_license>
XYZ:MusicManager User Fixed
```

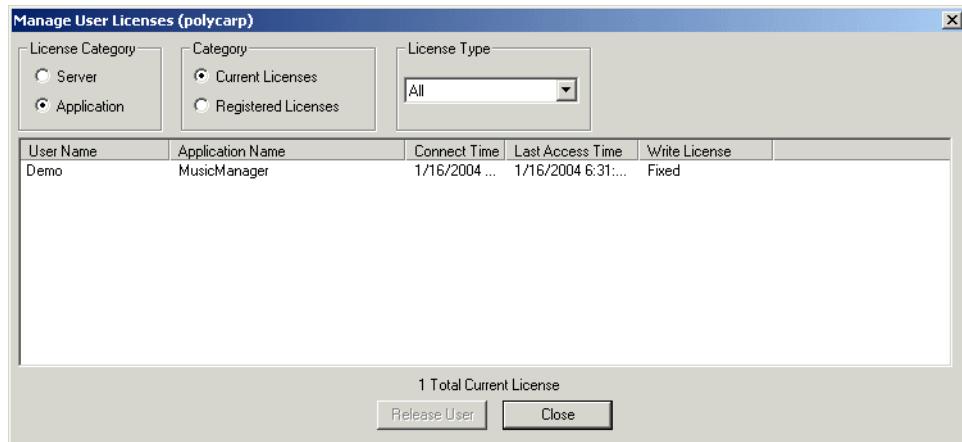
Separate multiple licenses with semicolons:

```
XYZ:MusicManager User Fixed; XYZ:NoiseManager User Fixed
```

- 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 BMC Remedy Administrator, as shown in Figure 23-4.

Figure 23-4: License information

For more information about managing licenses, see the *Configuring* guide.

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 376)
- Set operation type (page 377)
- Get operation type (page 378)
- XPATH function (page 379)
- Setting the starting record and setting the maximum limit (page 381)

For additional information about web services, see Chapter 6, “Web services.”

Create operation type

External applications can use the Create operation type to submit new entries into AR System.

When AR System receives an incoming request of type Create, it performs the following procedures:

- It parses the incoming XML code based on the input mapping and generates field values.
- It creates a new entry in the base form with these field values.
- This new entry creation triggers the OnSubmit filters.
- After the Create action is completely successful, AR System obtains field values from the newly created entry.
- The action of obtaining the field values triggers the OnGetEntry filters.
- AR System 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 will have the incoming data, but the output mapping will have 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 if you want to make different operations available on the same base form to accommodate different connecting applications.

Set operation type

External applications can use the Set operation of the web service to modify an existing entry in the AR System.

Unlike Create, the Set operation needs a qualification to identify the entry to modify. This qualification has to be specified in BMC Remedy Administrator 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’ = “0000000000000001” which means that any incoming request will 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 it allows XPATH expressions in its values, for example
‘RequestID’ = XPATH(“/ROOT/RequestID”)
- A completely dynamic qualification is a single XPATH expression, for example XPATH(“/ROOT/QueryString”).

For an explanation of XPATH expressions, see “XPATH function” on page 379.

When AR System receives an incoming request of the type Set, it performs the following actions:

- It determines if the qualification is dynamic; if so, it expands the XPATH expressions with data from the incoming XML to make the qualification static.
- It searches the base form for an entry matching the qualification.
- It 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 completely successful, AR System obtains field values from the recently modified entry.
- The action of obtaining the field values triggers the OnGetEntry filters.
- The AR System then 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 will have the incoming data, but the output mapping will have computed data, for example fields that are set by filters.

The `Set` operation type is similar to modifying an entry in the user tool and then clicking refresh to get back the same entry. All the rules for modify also apply—required values cannot be nulled out, 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.

For more information, see “The `Set` operation type for complex documents” on page 389.

Get operation type

External applications can use a `Get` operation to get details about an entry in AR System.

Like `Set`, the `Get` operation also needs a qualification that has to be specified in BMC Remedy Administrator. You cannot use an attachment field as a field reference in a qualification. For the `Get` operation, a qualification is automatically generated for you (this can be modified) and a particular entry is retrieved. For the `GetList` operation, you need to manually enter a qualification, or the system will retrieve all the entries. The qualification for both operations can be static, semidynamic, or completely dynamic.

When AR System receives an incoming request of type `Get`, it performs the following procedures:

- It determines if the qualification is dynamic. If so, it expands the `XPATH` expressions with data from the incoming XML to make the qualification static.
- It 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 only used for expanding `XPATH` expressions. AR System gets field values from the matched entries. For an explanation of `XPATH` expressions, see “`XPATH` function” on page 379.
- The action of obtaining the field values triggers the `OnGetEntry` filters.

AR System then 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 will ignore it.

The default operations listed are OpGet and OpGetList. These are merely names for the operation type Get. Use the mappings to create a Get or a GetAll operation. Get returns one entry, and GetAll returns multiple entries. For GetAll operations, map the form to a complex type with maxOccurs=(a number greater than 1 or unbounded) so that the resulting records (>1) can be passed on to the user. See “The get operation type for complex documents” on page 390 for more information.

XPATH function

When publishing web services 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 OpGet, OpGetList, and OpSet 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 only for web service qualifications. Access the available XPATH expressions by clicking the arrow on the right side of the Qualification bar in the Web Service dialog box and select XPath (see Figure A-1).

Figure A-1: XPath selection for qualification bar



An XPATH expression is a way of identifying an XML element inside an XML document. Its syntax is similar to a directory path syntax. Remember these guidelines as you create XPATH expressions:

- 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 only exist in the XML schema and be used only for the purpose of 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 will be substituted based on the XML data type.
- For strings, extra double quotes around the value are added by AR System.
- 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, then 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 you want 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 that you want to search on.

Here 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>2004-01-01T00:00:00.000000-08:00</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’=11 2	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’=10098 72000	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’>10 0	Qualification is of type: string.

For information about the XPATH specification, go to <http://www.w3.org/TR/xpath>.

Setting the starting record and setting the maximum limit



If you select the OpGetList 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 will have an XPATH expression that will retrieve 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 will be displayed.

B

Mapping web service data

This section describes the tools and processes used in mapping web service data.

The following topics are provided:

- Mapping to simple and complex documents (page 384)
- XML editing (page 397)
- Data types (page 408)

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.

Simple documents

With a simple XML document, the fields from one form are mapped to the XML element names. It is a simple list of ARFieldName/XMLElementName pairs.

► To map to simple documents

- 1 In the Create Web Service or the Create Filter dialog box, click the Input Mapping or Output Mapping button according to the mapping you want to set.

The Mapping dialog box appears together with the Object Properties dialog box as shown in the Figure B-1 on page 385. (For using a web service, the Form and XML Schema panes are reversed.)

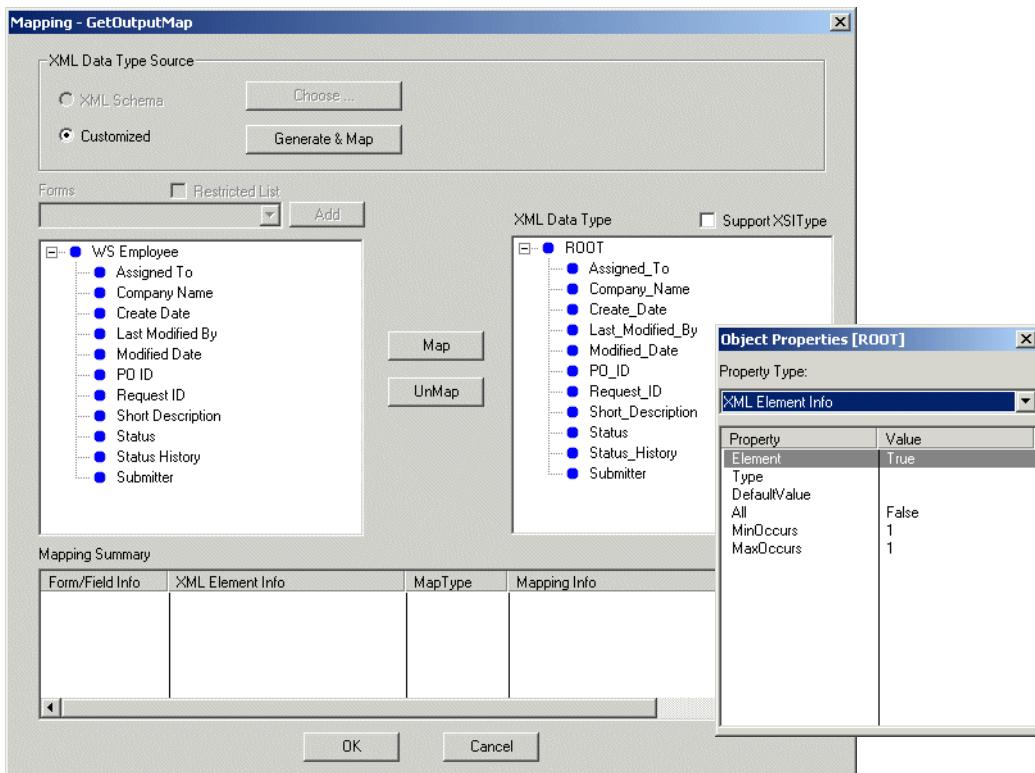
The Object Properties dialog box displays properties of a selected object. You can also display the Object Properties dialog box by right-clicking the element name in the XML Schema pane and choosing Properties.

Items with a solid circle are mapped, those with a hollow circle are not mapped. You can only map items of the same data type. See “Data types” on page 408 for more information.

If you left the XML Schema field blank in the Web Service dialog box, the System Generated option will be automatically selected.

- 2 Click the Generate & Map button and the AR System will generate an XML schema for you and automatically set the input and output mappings.

Figure B-1: Mapping and object properties dialog boxes (simple documents)



The Forms field arrow is for adding more forms to create parent and child relationships for mapping to complex XML Schemas. See “Complex hierarchical documents” on page 386.

The base form name and fields are displayed in the Forms pane, and the XML element names are displayed in the XML Schema pane.

- 3 If you opted for a system generated XML schema, your base form will be mapped to the ROOT element in your XML Schema, and the fields in your form will automatically be mapped to element names in the XML document.
- 4 Make any changes to the mapping, or the form, fields and elements in your XML schema.

► To remove an existing mapping

- 1 Select the field in the Forms pane or an element in the XML Data Type pane.
- 2 Click the UnMap button.

► To map a field to an XML element name

- 1 Select the unmapped field in the Form pane.
- 2 Select an available element name in the XML Data Type pane.
- 3 Click the Map button.

To add and remove fields, see “To add or remove an existing field from the form list” on page 404 and to modify the XML schema, see “To add a new element or attribute” on page 398 and “To cut, copy, or paste an element or attribute” on page 398.

- 4 Click OK to close the Mapping dialog box.
- 5 Save your web service.

Complex hierarchical documents

For complex hierarchical documents, the XML schema maps to multiple interrelated forms. In the following example, a purchase order XML is mapped to two forms, PurchaseOrder and LineItem. The data can be represented by the following figure:

PurchaseOrder			LineItem				
	Request ID	Description	Date	Request ID	Ln ID	Description	PO ID
007	XYZ Corp	2/12/04		0015	1	Memory	007
				0016	2	CPU	007
				0017	3	HardDisk	007
012	ABC, Inc.	3/01/04		0020	1	Scanner	012
				0021	2	Printer	012

```

graph LR
    PO[PurchaseOrder] -- Request ID --> LI[LineItem]
    PO -- Date --> LI -- Ln ID --
  
```

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 PurchaseOrder and LineItem are related through the ID fields:

- The PurchaseOrder form’s Request ID. This is the primary key in the parent form and it is unique. This is the key that establishes the relationship with the LineItem form’s foreign key.

- LineItem form's POID. This is the foreign key in the child form; it establishes the relationship with the PurchaseOrder form's primary key.
- LineItem form's Request ID. This is the primary key in the child form and is unique.
- LineItem form's Line ID. This is unique only within the subset of requests that reference the same PurchaseOrder. The LineID together with the POID form a “unique key.”

The XML input document in the example could be 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 the 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, then that is used externally as well. In that case, the request ID field will probably be 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 present in the XML document are the LineIDs of the child form. These LineIDs can be numbers, or strings, such as the description. The LineIDs are only used 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.

Nillable attributes

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. For more information about nillable attributes, see “Object properties” on page 398.

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 LineIDs of the existing three line items with the Line IDs of the two new line items. The following rules apply:

- If there are matching LineIDs, 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 LineID does not exist in the database, the server creates a new record in the line item form.
- If there is a missing LineID—that is, 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 that you choose in BMC Remedy Administrator when you are creating your Set operation type.
 - If you select Full from the list in the Composite Option field in the Set Query Options dialog box (see <need new figure xref>), the server will delete missing line items.
 - If you select Partial from the list in the Composite Option field in the Set Query Options dialog box (see <need new figure xref>), the server will ignore 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 LineID is unique within the scope of one document. The server will not be able to distinguish between duplicate LineIDs when performing a modify action. If this happens, results might not be as expected.

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 are generating 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 will be 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 only sets the item that has been sent.

For further information, see “Web services limitations” on page 110.

The Set operation type for complex documents

The Set (or Modify) operation for complex documents is more complicated than that for the simple document. In a simple document, your base form might have ten fields, all mapped to XML elements. If your modify request XML is returned with all ten XML elements, the server will modify 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 have specified MinOccurs=0 in the Object Properties dialog box for the XML elements, the server will modify only the fields in the input request. If your modify request XML is returned with fewer than ten elements, and you have set MinOccurs to equal other than 0, you will receive an error message.

The get operation type for complex documents

For GetAll (GetList) operations, map the form to a complex type with maxOccurs=(a number greater than 1 or unbounded) so that the resulting records (>1) can be passed on 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.

Filter flow for complex documents

When a complex document, such as the “PurchaseOrder” on page 386, is received by the AR System server, the AR System server updates the LineItem form by generating Push Field actions on the Purchase Order form for every line item in the Purchase Order 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 need to have created your forms and created a complex XML Schema. See “Advanced XML editing” on page 404 for more information. Choose the parent form as the base form of the web service and additional forms are used as child forms. A child form should not be used with more than one parent form. Also, to publish a web service, enter an XML Schema in the Web Services dialog box, and click Load.

► To map to complex documents

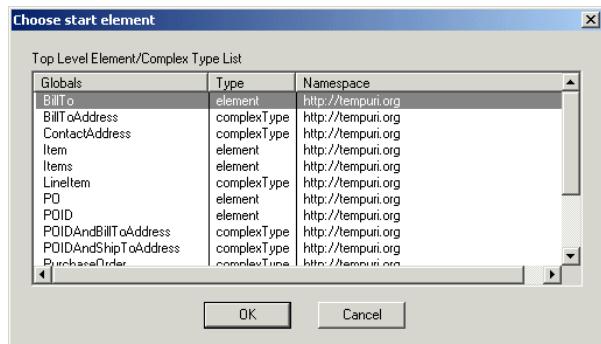
- 1 In the Create Web Service or the Create Filter dialog box, click the Input Mapping button or the Output Mapping button according to the mapping you want to set.

The Mapping dialog box appears.

- 2 Select an external XML schema or a system generated one.
 - a If you want the AR System to automatically generate an XML schema, click System Generated and then Generate and Map.
 - b If you have selected an external XML schema, select the XML Schema option button.
 - c Select Support XSIType, if necessary. If this option is checked, the system specifies xsiType for all XML data while creating XML documents.
 - d Click the Choose button.
 - e Click OK at the warning message that your existing mappings will be replaced.

The Choose start element dialog box is displayed showing complex types and elements.

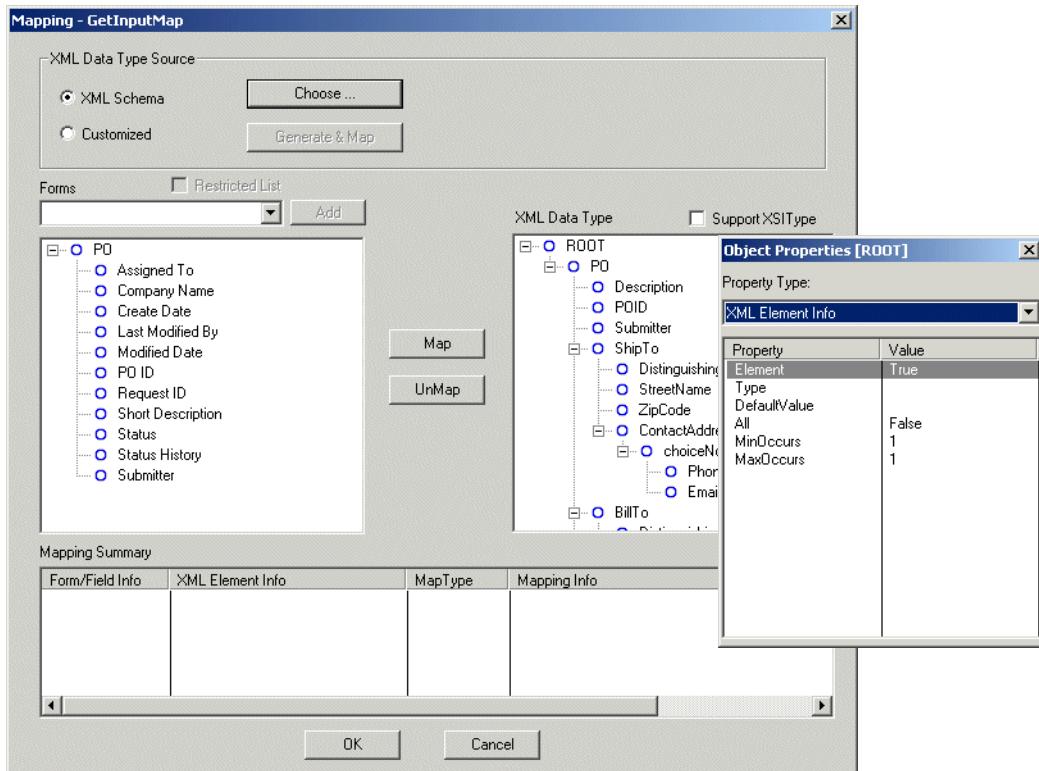
Figure B-2: Choose start element dialog box



- f Choose your start element, or start complex type. See “Importing an external XML schema” on page 405 for more information.

Your XML elements and complex types will be displayed in the XML Schema pane and your parent form will be displayed in the Forms pane as shown in Figure B-3. (For consuming a web service, the Form and XML Schema panes are reversed.)

Figure B-3: Mapping and object properties dialog boxes (complex documents)

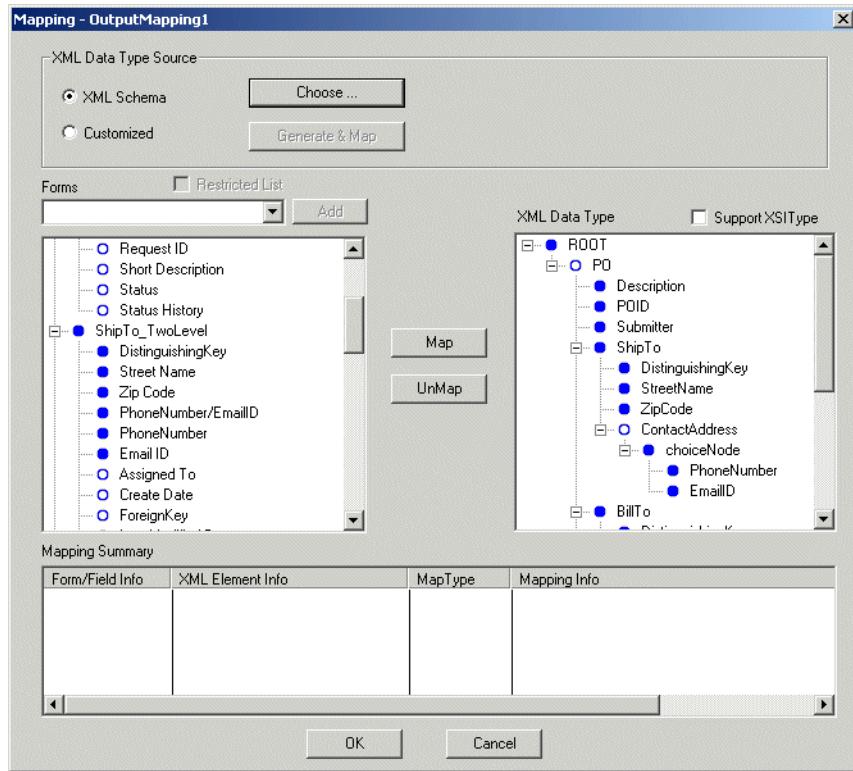


The Object Properties dialog box displays properties of a selected object. You can also display the Object Properties dialog box by right clicking the element name in the XML Schema pane and choosing Properties.

Items with a solid circle are mapped; those with a hollow circle are not mapped.

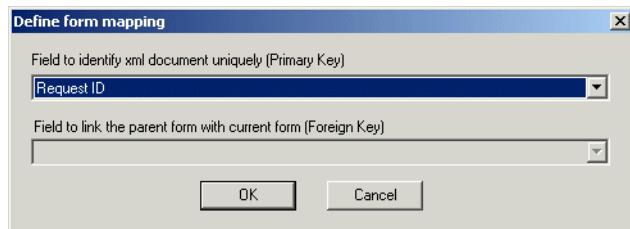
- 3 Add additional forms to the list using the Forms menu. These will be child forms.

All the selected forms and their fields will be listed in the Forms pane.

Figure B-4: Mapping dialog box showing mapping of forms, fields, and elements

- 4 Select the parent form name from the Forms pane.
- 5 Select the ROOT element in XML Schema pane to map the parent form. The parent form is typically mapped to the ROOT element.

The Define form mapping dialog box appears as shown in the following figure. Only character fields are listed that have been specified as Unique in the Indexes tab of the Form Properties dialog box.

Figure B-5: Define form mapping dialog box

- 6 Select the field from the list 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 foreign key field is disabled for the parent form.

- 7 Click OK to close the Define form mapping dialog box.
- 8 Map any other the fields in the parent form to elements in the XML schema.

Note: 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 408 for more information.

- 9 Map a child form in your forms list to elements in the XML schema list.

The Define form mapping: Establishing master detail relation dialog box appears.

Figure B-6: Define form Mapping:Establishing master detail relation



- 10 Select a value in the Field to distinguish this detail item from others field. This is the value that uniquely identifies each of the detail items in the child form of any given parent entry.

- 11 Select a value in the Field to link the parent form with the current form (Foreign Key) field. In our purchase example, it could be the POID.

The combination of distinguishing key and foreign key should uniquely identify an entry in a child form. BMC Remedy Administrator does not enforce this, but the AR System server will return an error if it detects non-compliance.

A field specified as foreign key should not be used in any field mapping since this field is used by system to store the foreign key. If it is mapped, that mapped XML element's value is usually overridden by foreign key value but this might not happen in all cases.

- 12 Map any other fields in the child form to XML elements.
- 13 Repeat step 9 to step 12 to map any other required 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, first map the choice node in the XML schema to the choice field in a form before mapping the choices. In Figure B-4 on page 393, for example, you would map PhoneNumber/EmailID field to the choiceNode element, then map the Phone Number field to the PhoneNumber element and the Email ID field to the EmailID element.

- 14 Click OK to close the Mapping dialog box.
- 15 Save your web service.

For further information, see “Web services limitations” on page 110.

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.

Since 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 returned.

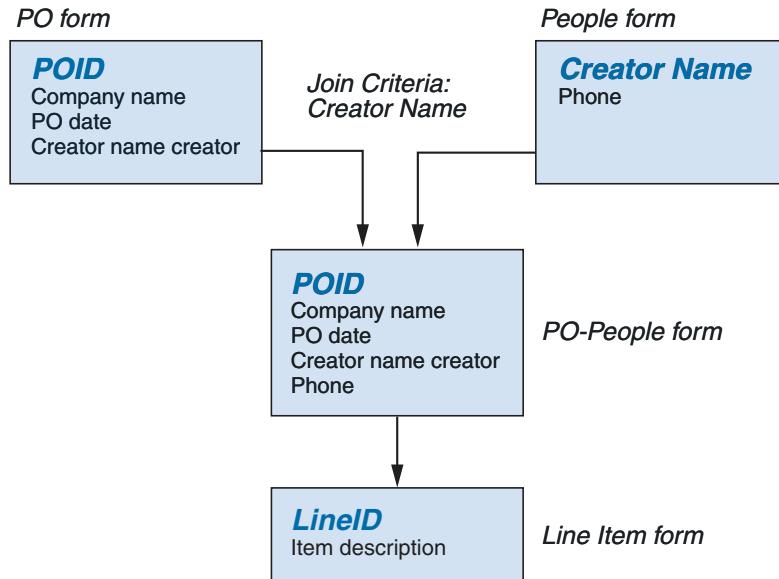
Example

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)

Since the Unique key POID will also be unique in the join form, choose it as the Primary key.

Figure B-7: Join form in web services



XML editing

You can perform simple editing tasks with BMC Remedy Administrator or, for more complex documents, you can create and edit an XML schema outside the AR System and import it. The following sections provide information about simple and complex XML editing.

Simple XML editing

BMC Remedy Administrator 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 pane to display a list of actions.
- 2 Choose New > Element or New > Attribute.

The new element or attribute will appear one level below the selected item.

- 3 Enter the name of your new element or attribute.

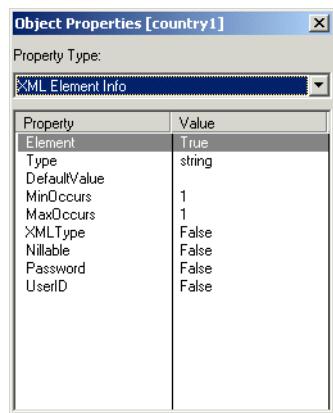
► To cut, copy, or paste an element or attribute

- 1 Right-click an element in the XML Data Type pane to display a list of actions.
- 2 Select the required action.
- 3 Paste to enter the new item one level below the item selected.

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 using the Object Properties dialog box as shown in the following figure.

Figure B-8: Object properties dialog box



To edit a property value, choose XML Element Info in the Property Type menu list, click the item in the value column corresponding to the property you want to edit and either select or type the value.

- **Element**—If the value for Element is set to “True,” the object is an element. If the value is set to “False,” the object is an attribute. ROOT is an element and cannot be edited.
- **XmlType**—Any leaf node (element or attribute) can be identified as xmlType. This allows the data to be treated as XML string data and AR System does not treat it like a regular string (that is, the data are not encoded or decoded).
- **MinOccurs**—This indicates number of instances of the element. By default it is 1. If it is zero, the element might or might not be present.
- **MaxOccurs**—This indicates maximum number of instances of the element. By default it is 1; 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.
- **DefaultValue**—This is the default value of an element unless it is overridden by the value in the actual document (incoming or outgoing).
- **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 that follow.

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.

Incoming XML elements

	<code>minOccurs=0 and nillable=false</code>	<code>minOccurs=0 and nillable=true</code>	<code>minOccurs=1 and nillable=false</code>	<code>minOccurs=1 and nillable=true</code>
Missing <code><name></code>	Name is not modified or set to AR default. ¹	Name is not modified or set to AR default. ¹	This is invalid XML. ²	This is invalid XML. ²
<code><name></name></code> OR <code><name/></code>	Name=\$NULL\$ or xsd default ³	Name=\$NULL\$ or xsd default ³	Name=\$NULL\$ or xsd default ³	Name=\$NULL\$ or xsd default ³
<code><name xsi:nil="true"></code> <code></name></code> OR <code><name xsi:nil="true"/></code>	This is invalid XML. ⁵	Name=\$NULL\$ ⁴	This is 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 that the XML element is mapped to 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` being 1, it is invalid XML. The client should not send such an XML packet. But if it does, AR System displays an error.

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, not the AR System default.) 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, and 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 not modified or set to AR default. ¹	This is invalid XML. ²
name=""	Name=\$NULL\$ ³	Name=\$NULL\$ ³

¹If 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, it 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.

²If 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 will display an error.

³If 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/> ³	<name/> ⁴	<name/> ⁴	<name/> ⁴
<name> is not mapped	Missing name ⁵	Missing name ⁵	This is invalid XML. ⁶	This is invalid XML. ⁶

¹If a field is null, AR System generates the XML as xsi:nil=true. However it can do so only if nillable=true.

²If nillable is false, AR System doesn't generate the element at all for null fields. However it can do so only if minOccurs=0.

³If nillable is false and minOccurs=1, AR System generates an element with empty content.

⁴If a field is of character type and it contains an empty string (this is extremely unusual, it can be done only through the driver program), AR System generates an element with empty content.

⁵If an XML element is not mapped to a field, AR System does not generate an element for that.

⁶If an XML element is not mapped to a field, but it has minOccurs=1, AR System does not generate an element for that, which means that the XML generated by AR System is invalid.

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 ³	Invalid XML ⁴

¹If a field is null, AR System generates an attribute with empty content.

²If a field is a character type and it contains an empty string (this is rare; it can be done only through the driver program), AR System generates an attribute with empty content

³If an XML attribute is not mapped to a field, AR System does not generate an attribute for it.

⁴If an XML attribute is not mapped to a field, but has use=required, AR System does not generate an attribute for it, therefore the XML generated by AR System is invalid.

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 only be mapped 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 till 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 on the Generate Schema button on the Mapping dialog box. This will create XML elements that are automatically mapped.

► **To remove an existing field from the mapping list**

- 1 In the Mapping dialog box, right-click on the field name.
- 2 Select Remove Field from the drop-down list.

► **To add or remove an existing field from the form list**

- 1 Right-click on the form name.
- 2 Select Add Fields or Remove Fields from the drop-down list.
- 3 Select the fields to remove from the Remove Fields list, or select the fields to add from the Add Fields list.

Even if you have removed a field from the list and generated the schema, the field will still be listed the next time you open the web service because it still exists on the form. The field will display an empty circle next to its name to indicate that it is not mapped. However, if you delete an element or attribute from the XML schema list, it is deleted from the XML schema and will not reappear until you create a new entry.

XML editing is allowed only while you are publishing a web service from the AR System. When you are consuming an external web service, the XML format is decided by the external web service and you must conform to it. BMC Remedy Administrator disables all editing features in that case. You can only choose which fields you want to map to which XML elements.

XML documents can specify the data type within the document itself instead of in the XML schema. If you want the output document to contain xsiType information so that the consumers of the web service can process the document correctly, check the Support xsiType option in the Mapping dialog box. In this case, the system generates xsiType information.

Advanced XML editing

The Mapping dialog box allows only simple XML editing. For complex editing, you need to 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 using a simple text editor.

For more information, an online tutorial on XML schemas is available from:
<http://www.xfront.com/index.html#schema>.

Once you have designed an XML schema, you can import it into BMC Remedy Administrator. However once you import it, all the XML editing features in BMC Remedy Administrator are disabled. Consequently, you can either edit the XML completely inside BMC Remedy Administrator, or completely externally.

If you are using 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`. If you use an older version of an XSD file, your mappings might not be as expected.

Importing an external XML schema

Only 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 if you want to use multiple XML schemas, create a new XML schema that includes or imports all of the schemas you want to use.

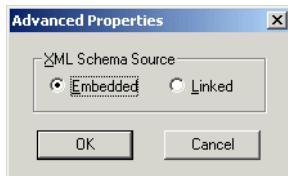
► To import external XML schema

- 1 Specify your XML Schema (XSD file) in the XML Schema field

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 which 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 will be referenced in your generated WSDL file, so your path must be accessible over the network. You should make a web server directory to hold your XSD files, and enter the http path to this directory in the Web Services dialog box.

- 2 Click the Load button.
- 3 Click OK to accept the loss of your existing mappings, or you might see a confirmation message.
- 4 Click the Options button to display the Advanced Properties dialog box as shown in the following figure.

Figure B-9: Advanced properties dialog box

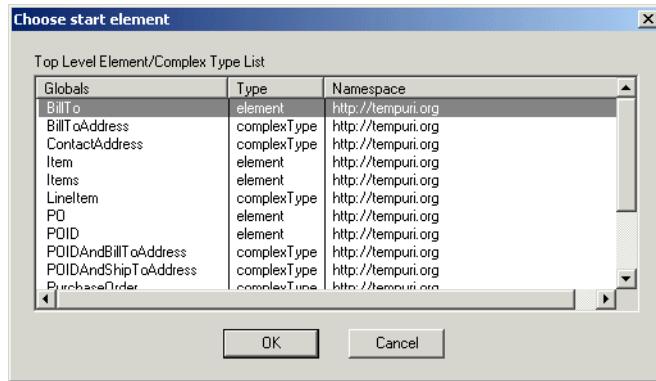
- a** Choose Embedded if you had entered a local file system path for your XSD. In this case, AR System loads the specified XML schema and all dependent XML schemas. It stores the entire XSD file and all other files that the XSD file includes or imports. The WSDL also has all the XSDs embedded in the types section. This is the default option.
- b** You can choose Linked if you entered a network accessible path (http or ftp) for the XSD. In this case, AR System does not store the XSD files, but stores a reference to the specified schema in the web service object as well as 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 Input Mapping dialog box, select XML Schema and then click Choose.
- 6 Select a start element.

The Choose start element dialog box appears. Separate mappings can be based on separate global elements, but all of them must come from the same XML schema. BMC Remedy Administrator displays a list of global elements and global complex types either present directly in your XSD file, or those included or imported into it.

Figure B-10: Choose start element dialog box

If you choose an element, the element and all its successors will be added as a child of the ROOT element, whereas, if you choose a complexType, the contents of the complexType will be added as children 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 will update them for you. If there are no overlapping global or complex types, AR System will preserve the content of the original mapping.

BMC Remedy Administrator does not support all features of XML schemas, and when using a web service there are restrictions that apply to the external WSDL file. See the *Release Notes* for more information about limitations.

Data types

When mapping an XML element to an AR System field, BMC Remedy Administrator allows only compatible data types, both for consuming and publishing.

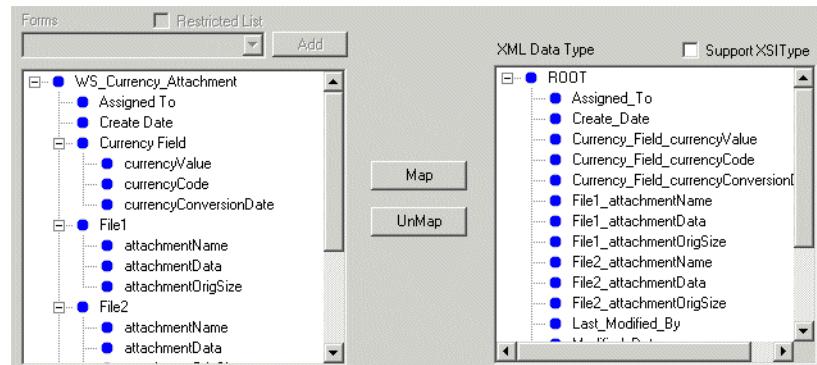
AR System data types	XML schema data types
CHAR	string, duration, anyURI, QName, NOTATION, normalizedString, token, language, NMTOKEN, Name, NCName, ID, IDREF, ENTITY, integer, nonPositiveInteger, negativeInteger, nonNegativeInteger, positiveInteger
ENUM	string
INTEGER	int, boolean, short, byte, unsignedInt, unsignedShort, unsignedByte
REAL	double, float
DIARY	string
DATE/TIME	dateTime
DECIMAL	decimal, long, unsignedLong
DATE	date, gYearMonth, gYear, gMonthDay, gDay, gMonth
	Defaults: YEAR - 1000 (leap year), month - 01, day - 01
TIME	time
CURRENCYcurrencyValue	decimal
CURRENCYcurrencyCode	string
CURRENCYcurrencyConversionDate	dateTime
STATUS HISTORY	string
ATTACHMENTName	string
ATTACHMENTData	base64Binary
ATTACHMENTOrigSize	int

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 are retrieving a diary field from the 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 are sending 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 the AR System.
- A currency field is treated as a collection of four parts, and each part needs to be mapped separately, as shown in Figure B-11.
- An attachment field is treated as a collection of three parts, as shown in Figure B-11.

Figure B-11: Form and XML schema panes in mapping dialog box



C Web service examples

This section provides examples of creating and consuming web services of various levels of complexity.

The following topics are provided:

- Example 1: publishing a simple flat document (page 412)
- Example 2: consuming a simple flat document (page 417)
- Example 3: publishing a complex document (page 423)
- Example 4: consuming a complex document (page 440)

For additional information about web services, see Chapter 6, “Web services.”

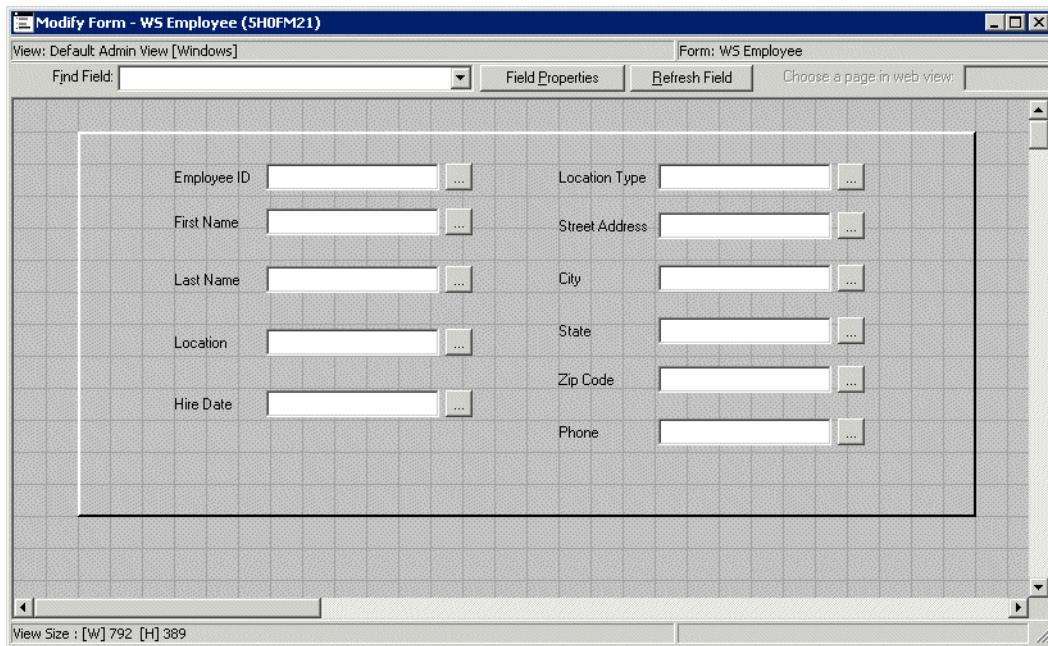
Example 1: publishing a simple flat document

In this example you will publish a web service that has the following default operations for an employee record: OpCreate, OpSet, OpGet, and OpGetAll.

► To publish a simple flat document

- 1 Create a form that displays your employee data. This will be the Base Form used in creating your web service. Figure C-1 shows a sample form. All of the fields are character fields.

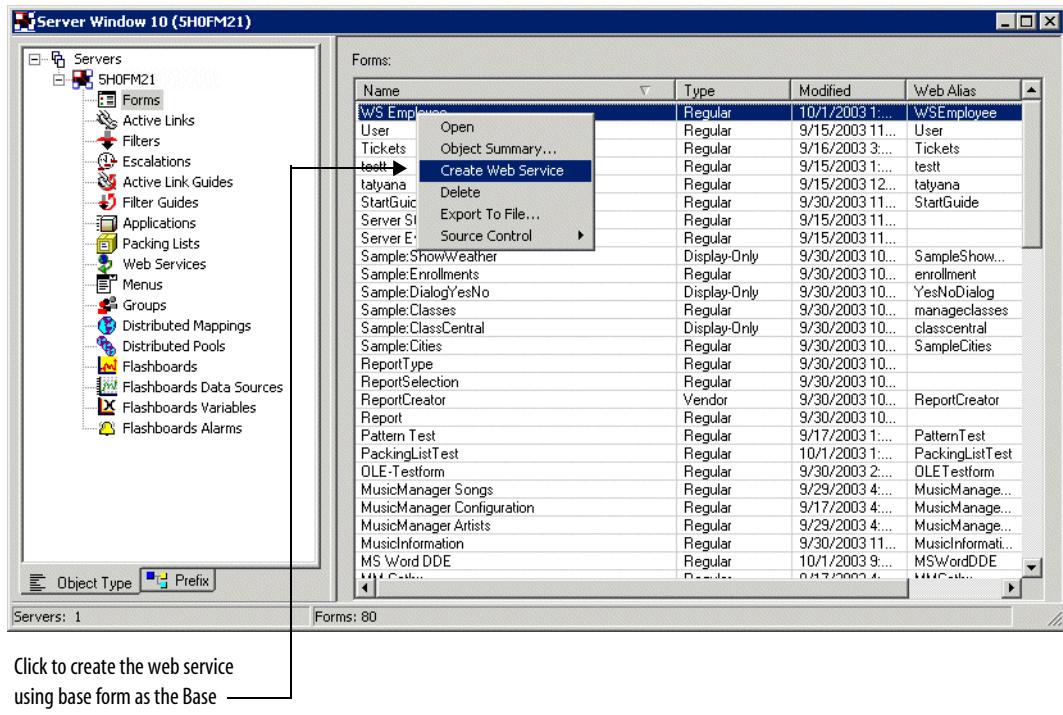
Figure C-1: Sample form for employee record



- 2 Right-click on the form name (WS Employee) in the server window as shown in the following figure.

- 3 Select Create Web Server from the menu.

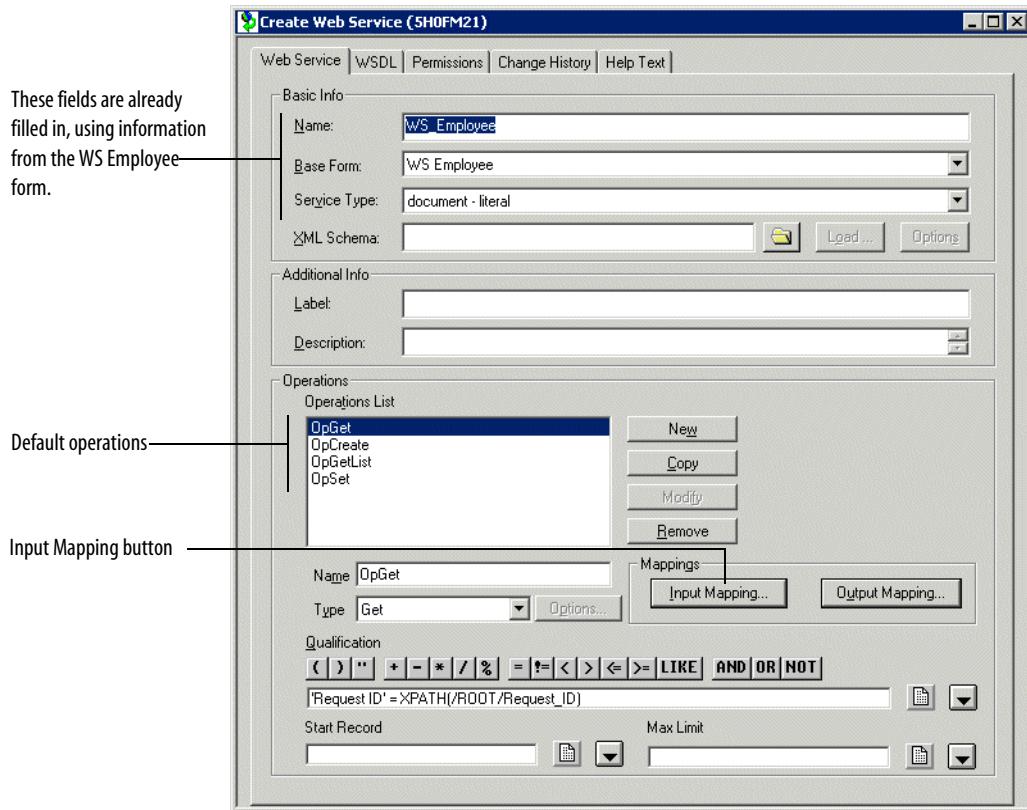
Figure C-2: Server window showing web service selection



Click to create the web service
using base form as the Base
Form.

The Create Web Service dialog box appears with the default settings as shown in Figure C-3.

Figure C-3: Create web service dialog box for employee web service



BMC Remedy Administrator automatically populates the Base Form field with your WS Employee form, the default service type is document literal, and the default web service name is the same as your base form name: WS Employee. The label and description fields are optional.

The default operations are automatically displayed in the Operations List.

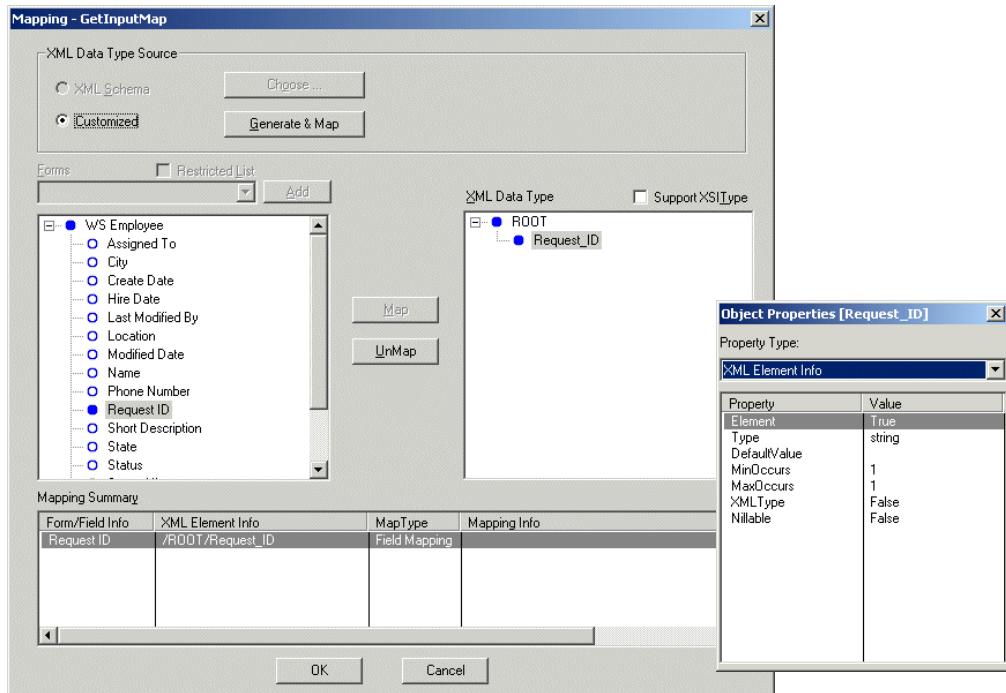
4 Select an operation from the Operations list.

The Name field will be automatically populated with the operation name, and the Type field will be automatically populated with the type of operation to be performed.

The fields on your WSEmployee form are mapped to XML-compliant element names in an XML schema.

- 5 Click the Input Mapping button in the Create Web Service dialog box to view these mappings. (You do not need to modify these mappings for this example.)

Figure C-4: Mapping dialog box for employee web service



A solid circle indicates that the field or XML element is mapped.

The Object Properties box refers to the XML properties of the XML elements.

- 6 Close the mapping dialog box.
- 7 Choose File > Save Web Service.
- 8 Click the WSDL tab on your Web Service dialog box. (See Figure C-3 on page 414.)
You will see a sample URL for your WSDL file displayed in the field.
- 9 Complete the URL as appropriate for your configuration, as follows:
 - a Replace *fs* with the name of the web server where the mid tier is running.

- b** After WSDL, add /public or /protected depending on the web service's permissions.

For example, if the mid tier server is POLYCARP and the web service has public permissions, use:

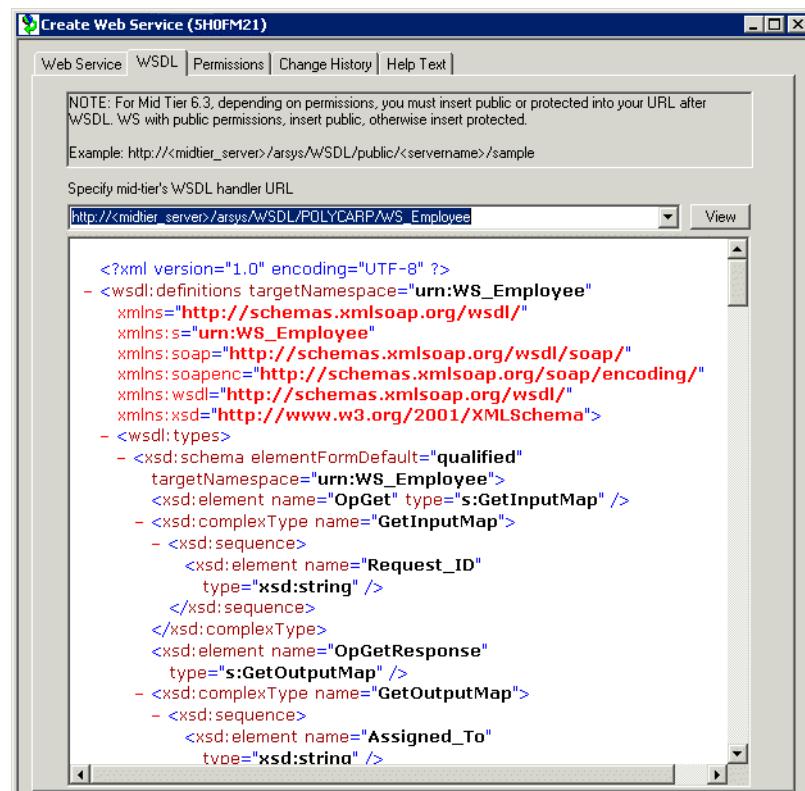
`http://POLYCARP/arsys/WSDL/public/POLYCARP/WS_Employee.`

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

`http://POLYCARP/arsys/WSDL/protected/POLYCARP/WS_Employee.`

- 10** Click the View button to view your WSDL file in the window.
- 11** Enter your name and password at the prompt, if necessary, and click Login.

Figure C-5: WSDL tab on web service dialog box for employee web service



- 12** Set the permissions to Public.

This is important if you are going to publish your web service over an internet or intranet for general use.

13 Save your web service.

Administrators with the appropriate SOAP protocol can now access this WSDL file with any browser.

Example 2: consuming a simple flat document

In this example you will access an external web service and, with the use of the Set Fields to Web Service filter, set the data into an AR System form. The external web service takes a stock ticker symbol as input and outputs the 20-minute delayed stock quote for that stock.

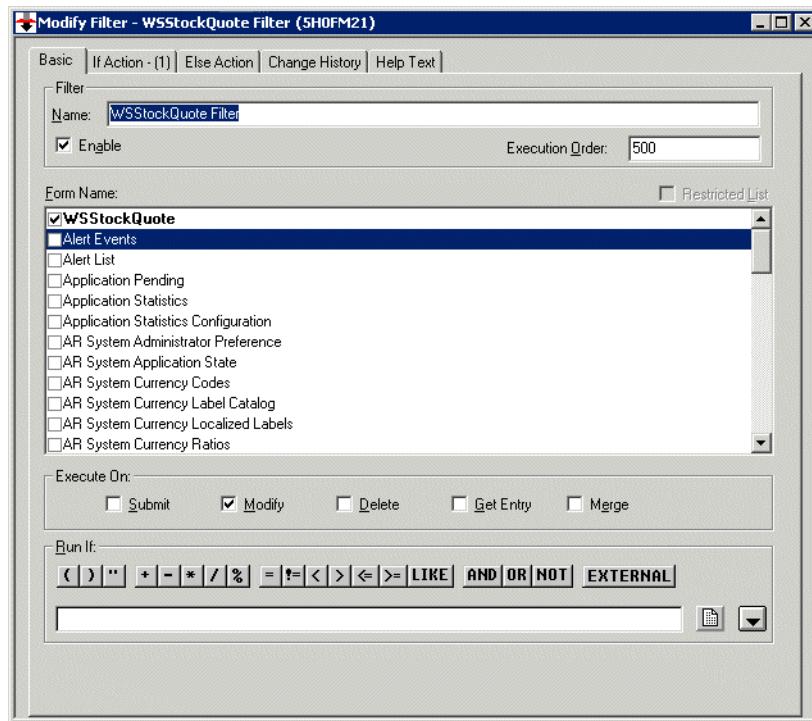
► To consume a simple flat document

- 1 In BMC Remedy Administrator, create a form; the one in our example is called WSStockQuote.
- 2 Add two fields:
 - A character field (called Stock Ticker Symbol in the example) in which you will enter the stock ticker symbol (for example, BMC).
 - A decimal field (called 20 minute delayed stock quote in the example) into which the quote for that stock will be set when information is received from the external web service.

Figure C-6: Form for stock quote web service

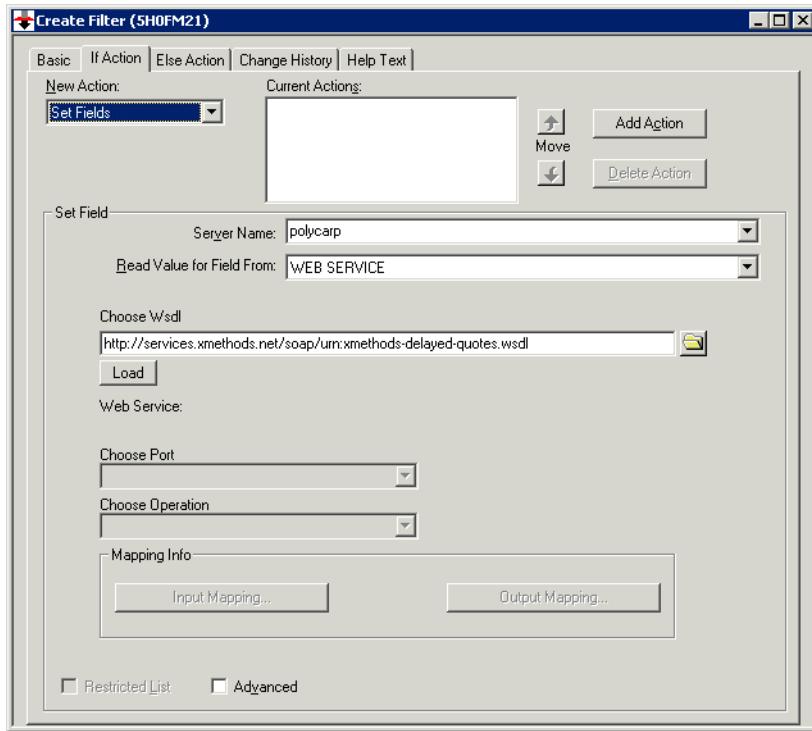
- 3 In the server window, choose File > New Server Object.
- 4 Click Filter and OK to display the Create Filter dialog box, Basic tab.
- 5 Enter a Name for the filter, such as WSStockQuote Filter.
- 6 Check the form WSStockQuote to use as the base form.
- 7 Check the Execute On Modify option button.

Figure C-7: Basic tab of WSStockQuote filter window



- 8 Click the If Action tab.
- 9 Select Set Fields from the New Action field.

Figure C-8: If action tab for the WSStockQuote filter window



- 10 In the Server Name field, enter the name of your server on which the Web Service filter plug-in is installed.
- 11 Select WEB SERVICE in the Read Value for Field From field.
- 12 Type the path to the WSDL file of your external web service in the Choose WSDL field:

`http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl`

Note: This WSDL reference might not work if its website is not functioning.

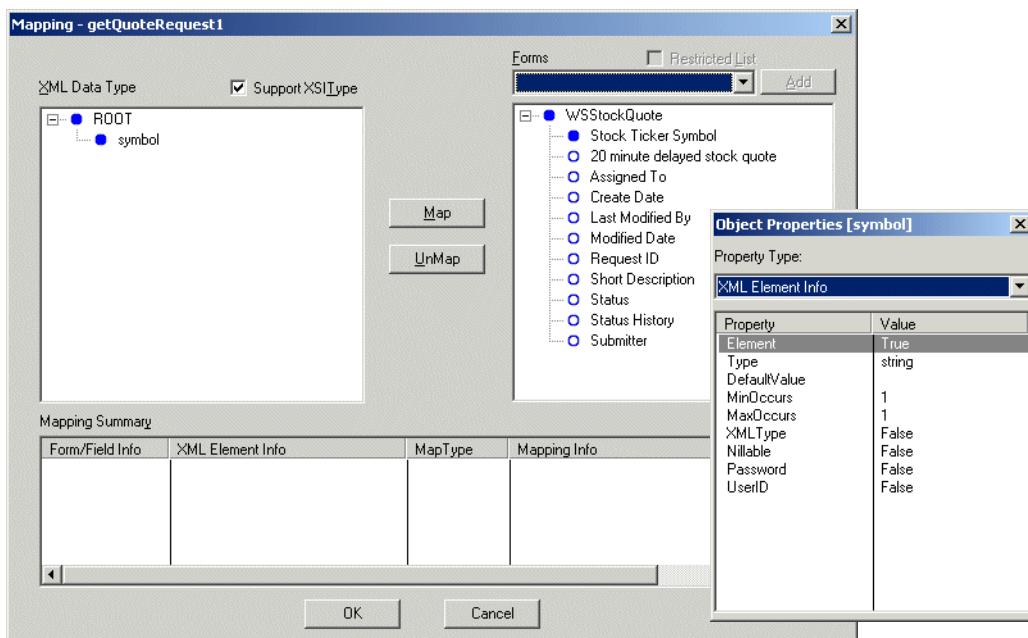
- 13 Click Load.
- 14 In the Choose Port field, select the port that the web service should use. The name of the web service (net.xmethods.services.stockquote.StockQuote) is displayed in the dialog box under the Web Service heading.

The available operations for the stock quote web service are automatically listed. In the example there is only one operation, but a web service might have multiple operations available.

- 15 Choose getQuote from the operations drop-down list.
- 16 Click the Input Mapping button to display the Mapping dialog box.
- 17 The ROOT element of the XML schema is automatically mapped to the WSStockQuote form.
- 18 Select the symbol element and the Stock Ticker Symbol field. You created this field in step 2 on page 417 to contain the stock symbol.
- 19 Click the Map button.

The symbol element (string type) is now mapped to the Stock Ticker Symbol field (field designed to contain a string).

Figure C-9: Input mapping for StockQuote web service

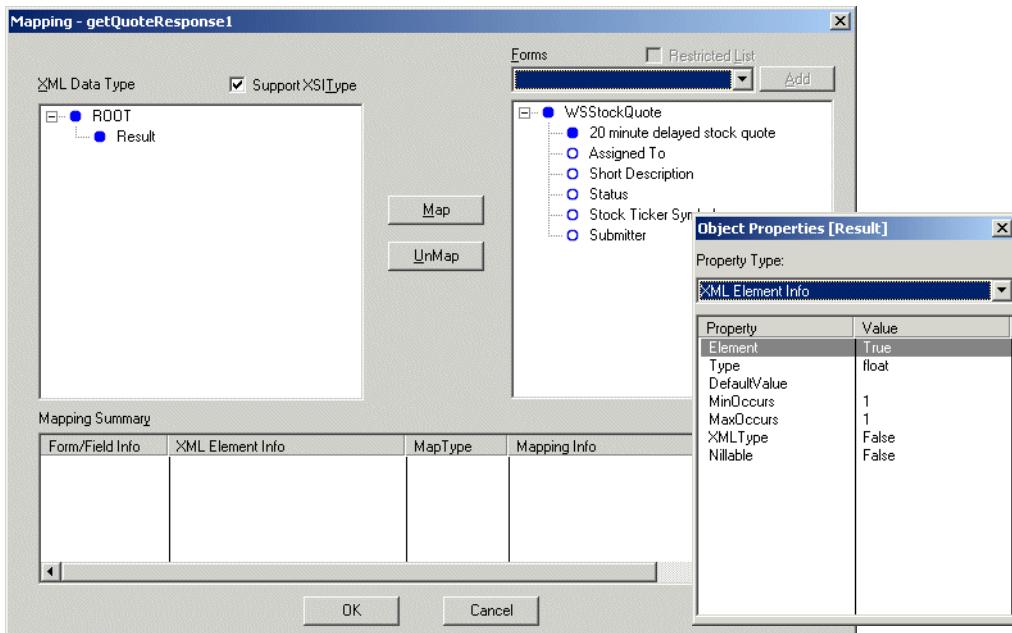


- 20 Click OK to close the Mapping dialog box.
- 21 Click the Output Mapping button to display the Mapping dialog box.

- 22 The ROOT element of the XML schema and the WSStockQuote form will be automatically mapped.
- 23 Select the Result element and the 20 minute delayed stock quote field as shown in the following figure. You created this field in step 2 to contain the stock quote.
- 24 Click the Map button.

The Result element (string type) is now mapped to the 20 minute delayed stock quote field (field designed to contain a string).

Figure C-10: Output mapping for StockQuote web service



- 25 Click OK to close the Mapping dialog box.
- 26 Click Add Action to add the action to the Current Actions pane. See “If action tab for the WSStockQuote filter window” on page 419.
- 27 Save the filter.
- 28 Open BMC Remedy User and log in.
- 29 Open the WSStockQuote form.
- 30 Enter data in the required fields.

31 Save your form.

32 Search to open the form in Modify mode.

You created the WSStockQuote Filter to trigger On Modify. See Figure C-7 on page 418.

33 Enter a stock value (for example, BMC) in the Stock Ticker Symbol field.

34 Click Save or Submit.

The quote for the stock you entered is displayed in the Stock Ticker Symbol field as shown in Figure C-11.

Figure C-11: WSStockQuote form in BMC Remedy User

The screenshot shows a Windows application window titled "WSStockQuote (Modify)". The title bar also displays "Matching WSStockQuote" and the time "0:06". The main area is titled "Modify WSStockQuote 0000000000000001". A blue header bar says "BMC - Remedy AR System". The form fields include:

- Request ID: 0000000000000001
- Stock Ticker Symbol: BMC
- Submitter: Ivan Stampovic
- Create Date: 10/2/2003 4:48:08 PM
- Last Modified By: Demo
- Modified Date: 10/2/2003 4:48:19 PM
- Status: New Assigned Fixed Rejected Closed
- Short Description: Eager stock quotes

A "Save" button is visible in the top right corner of the modify window.

35 Try different stock symbols.

Example 3: publishing a complex document

In this example you are going to publish a web service with two operations:

- CreatePurchaseOrder that takes PO information as input and returns the Request ID as output.
- GetPurchaseOrder that takes the PO ID as input and returns the information of that Purchase Order.

The process of this example for publishing a complex document is as follows:

Step 1 Create an .xsd file. (see page 423)

Step 2 Create the forms. (see page 424)

Step 3 Create a web service. (see page 428)

Step 4 Map the CreatePurchaseOrder operation. (see page 429)

Step 5 Map the GetPurchaseOrder operation. (see page 435)

Step 6 View your WSDL. (see page 439)

► To create an .xsd file

- 1 Create an XML schema (.xsd file) containing the elements for your Purchase Order.
- 2 Name the schema P0.xsd. An example follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by AR System
-->
<xss:schema targetNamespace="http://tempuri.org" xmlns:xss="http://
www.w3.org/2001/XMLSchema" xmlns="http://tempuri.org"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xss:element name="PO" type="PurchaseOrder">
        <xss:annotation>
            <xss:documentation>Comment describing your root element</
xss:documentation>
        </xss:annotation>
    </xss:element>
    <xss:complexType name="PurchaseOrder">
        <xss:sequence>
            <xss:element name="POID" type="xs:string"/>
            <xss:element name="CompanyName" type="xs:string"/>
            <xss:element name="Description" type="xs:string"/>
        </xss:sequence>
    </xss:complexType>
</xss:schema>
```

```
<xss:element name="PhoneNumber" type="xs:string"/>
<xss:element ref="Items"/>
</xss:sequence>
</xss:complexType>
<xss:complexType name="LineItem">
<xss:sequence>
<xss:element name="ItemName" type="xs:string"/>
<xss:element name="Quantity" type="xs:int"/>
<xss:element name="ItemId" type="xs:string"/>
</xss:sequence>
</xss:complexType>
<xss:element name="Item" type="LineItem"/>
<xss:element name="Items">
<xss:complexType>
<xss:sequence>
<xss:element ref="Item" minOccurs="0"
maxOccurs="unbounded"/>
</xss:sequence>
</xss:complexType>
</xss:element>
</xss:schema>
```

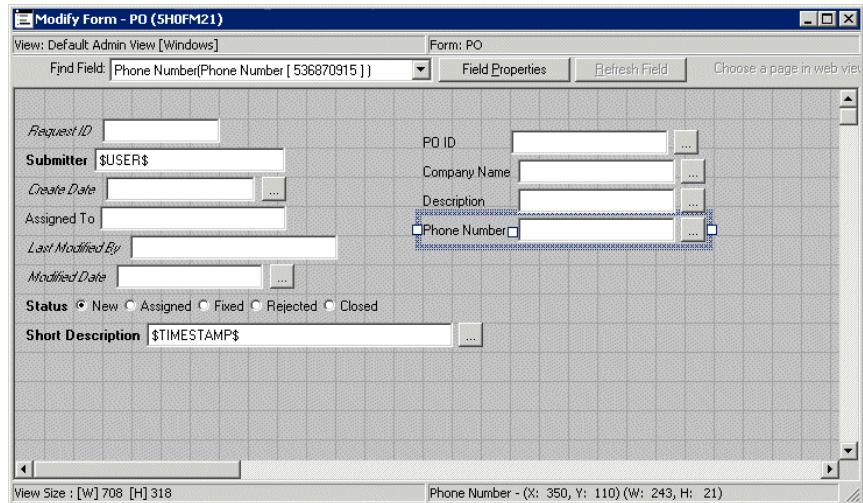
► To create your forms

- 1 Create the first of two forms.

Call the form PO and add four character fields, PO ID, Company Name, Description, and Phone Number. This will be the parent form.

- 2 (Optional) Hide all the other fields on the form.
- 3 Give a default value to Submitter and Short Description fields, since they are required fields.

Your form should look similar to the one shown in Figure C-12.

Figure C-12: PO form

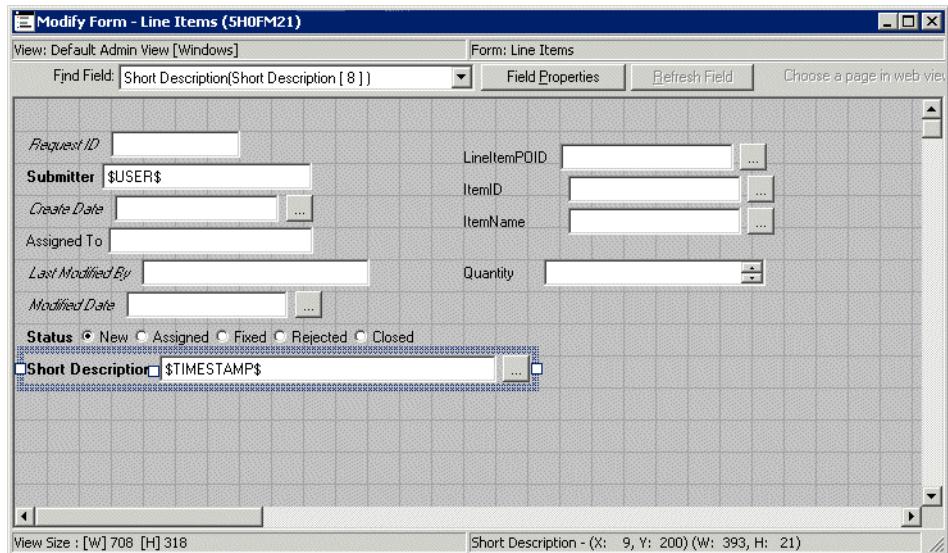
- 4 Choose Form > Form Properties from the Menu bar to open the Form Properties dialog box.
- 5 Click the Indexes tab.
- 6 Select PO ID, and add it to the Index On pane.
- 7 Check the Unique box.

In this example, we are using PO ID as the primary key. However, in most situations you would use Request ID as the primary key. In those situations, you do not need to create the PO ID field, and you do not need to perform step 4 to step 7.

- 8 Save the form.
- 9 Create the form named LineItems, which is the detail form, add character fields, LineItemPOID, ItemID, ItemName and an integer field Quantity. This LineItemPOID is going to be the foreign key and ItemID is going to be the distinguishing key.
- 10 (Optional) Hide all the other fields.
- 11 Set the Data Length of the LineItemPOID and ItemID fields to 40.
- 12 Give a default value to the required fields Submitter and Short Description (these are hidden).
- 13 Save the form.

Your form should look similar to the one shown in Figure C-13.

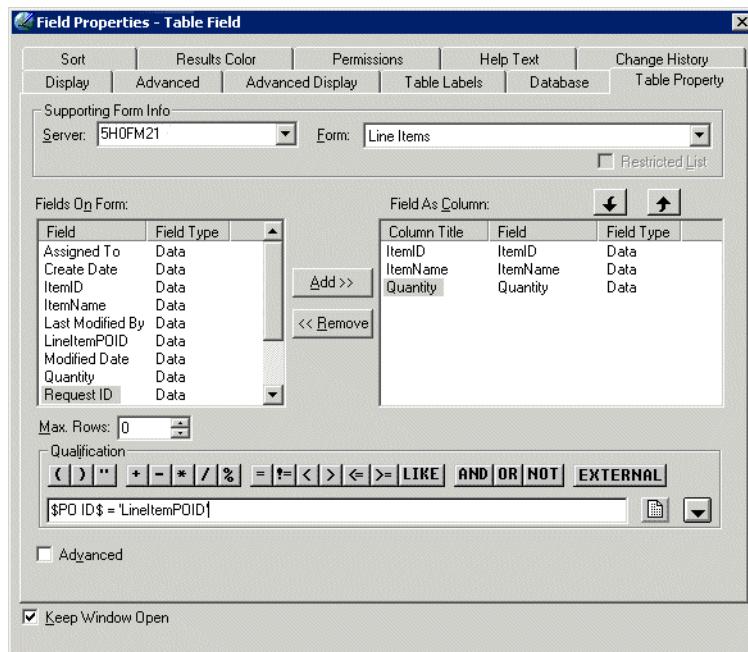
Figure C-13: LineItems form



- 14 Choose Form > Form Properties from the menu bar to open the Form Properties dialog box.
 - 15 Click the Indexes tab.
 - 16 Add LineItemPOID and ItemID to the Index On pane to make the combination of these two fields (LineItemPOID and ItemID) unique.
Making the distinguishing key and the foreign key unique maintains the consistency with the data, even if the data is submitted with the other tools on this form.
 - 17 Save the form.
- The following steps, step 18 to step 26 are optional. They will enable you to view the line items of the corresponding Purchase Order in the same form.
- 18 Open the PO form.
 - 19 Add a table field by right-clicking and select Table Field from the context menu.
 - 20 Double-click on the table field to open the Field Properties dialog box.
 - 21 Click the Table Property tab.

- 22 Select LineItems from the Forms list.
- 23 Add ItemID, Item Name and Quantity to the Field As Column pane.
- 24 Enter the qualification `$PO ID$ = 'LineItemPOID'` in the Qualification field. This ensures that the PO ID of the parent form PO matches the LineItemPOID of the LineItems form.

Figure C-14: Field properties dialog box—Table Property tab



If you had used Request ID as your primary key, enter this qualification instead:

`$Request ID$ = 'LineItemPOID'`

- 25 Close the Field Properties dialog box.

Your form should look similar to the one shown in Figure C-15.

Figure C-15: PO form with a table

The screenshot shows the 'Modify Form - PO (5H0FM21)' window. At the top, there are tabs for 'View: Default Admin View [Windows]' and 'Form: PO'. Below these are buttons for 'Find Field', 'Field Properties', 'Refresh Field', and 'Choose a page in web view'. The main area contains several input fields: 'Request ID', 'Submitter (\$USER\$)', 'Create Date', 'Assigned To', 'Last Modified By', 'Modified Date', 'Status' (radio buttons for New, Assigned, Fixed, Rejected, Closed), and 'Short Description' (\$TIMESTAMP\$). To the right of these fields is a table field with three columns: 'ItemID', 'ItemName', and 'Quantity'. The table has a header row and is currently empty. The bottom of the window shows 'View Size : [W] 789 [H] 369'.

26 Save your form and close it.

► To create your web service

- 1 Right-click on the parent form name (PO) in the Server Window (see Figure C-2 on page 413).
 - 2 Select Create Web Service.
- The Create Web Service dialog box appears.
- The AR System automatically populates the Name and the Base Form field with your parent PO form name.
- 3 Select document-literal from the Service Type field, if it is not selected.
 - 4 Enter the path, or browse to your `PO.xsd` file in the XML Schema field.
 - 5 Click the Load button, and OK at the confirmation dialog box.
 - 6 Enter the label `wsPO` in the Label field.
 - 7 Enter “Web Service to Create and Get a Purchase Order” in the Description field.
 - 8 Select `OpGetList` in the Operations List and click Remove.
 - 9 Select `OpSet` in the Operations List and click Remove.

- 10** Select OpCreate in the Operations List.

OpCreate appears in the Name field.

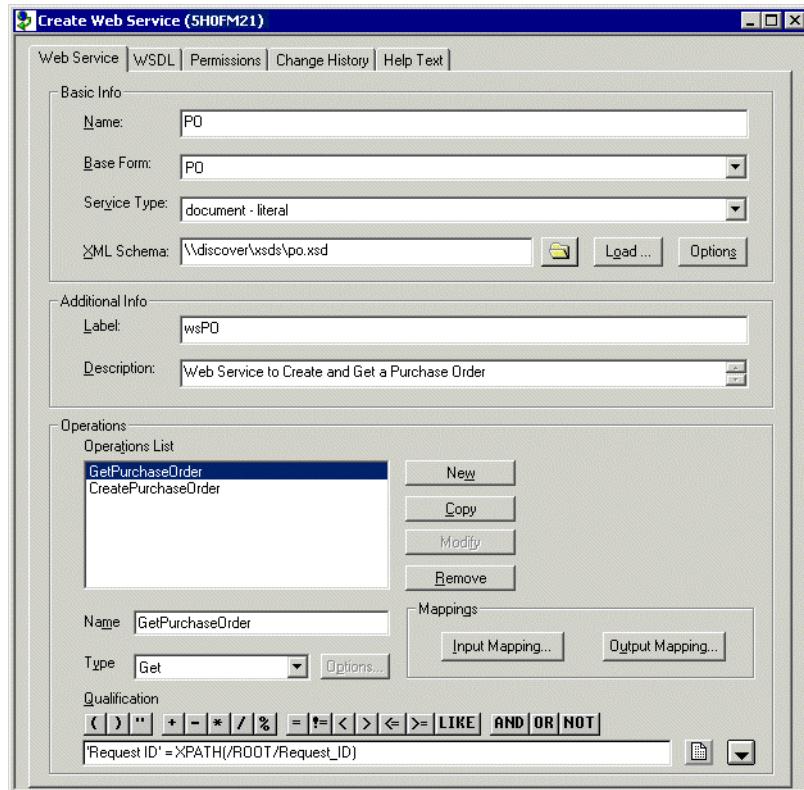
- 11** Replace OpCreate in the Name field with CreatePurchaseOrder.

- 12** Click Modify.

The new name appears in the Operations List.

- 13** Do the same with OpGet and rename it to GetPurchaseOrder.

Figure C-16: Create web service dialog box for purchase order



► **To map the CreatePurchaseOrder operation**

- 1** Select CreatePurchaseOrder in the Operations List.

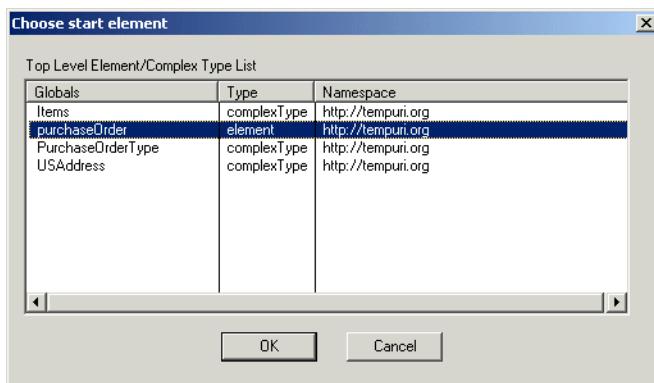
- 2** Click the Input Mapping button.

The Mapping dialog box appears.

- 3** Select the XML Schema option button.

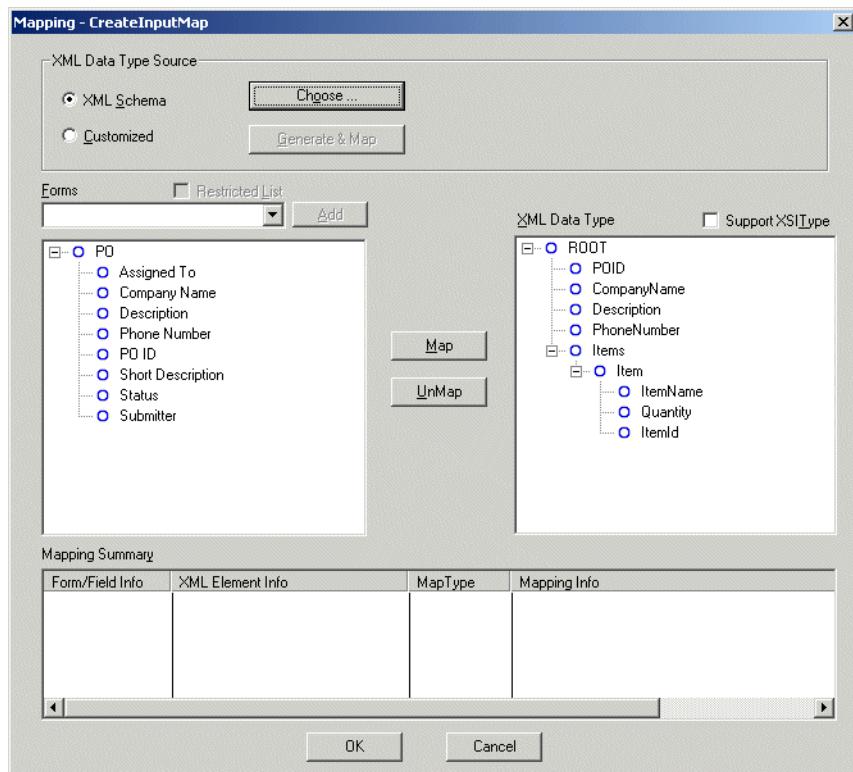
- 4 Click the Choose button.
- 5 Click OK at the warning message that your existing mappings will be replaced.
- 6 Choose purchaseOrder as the start element in the Choose start element dialog box, and click OK.

Figure C-17: Choose start element dialog box for create PO web service



In the Mapping dialog box, only PO form fields are displayed in the Forms pane, the XML Schema elements and complex types appear in the XML Schema pane, as shown in Figure C-18.

Figure C-18: Mapping dialog box for PO web service input



- 7 Click the selection arrow on the Forms field.
 - 8 From the Forms drop-down list, select the form LineItems and click Add. The form name and the fields appear below the parent form in the Forms pane.
 - 9 Click PO in the Forms list and ROOT in the XML Data Type list.
 - 10 Click Map.
- The Define form mapping dialog box appears.
- 11 Select PO ID as the primary key.

Figure C-19: Define form mapping dialog box for creating PO web service

- 12 Click OK to close the Define form mapping dialog box.
- 13 Map the other fields on the PO form to XML elements and complex types as shown in the following table

Forms and fields	XML elements and complex types	Field to identify xml document uniquely
PO	ROOT/PO	PO ID
PO ID	POID	
Description	Description	
Phone Number	PhoneNumber	
Company Name	CompanyName	

- 14 Select LineItems from the Forms pane and Item from the XML Data Type pane.
- 15 Click Map.

The Define Form Mapping:Establishing master detail relation form appears.

Figure C-20: Define form mapping: establishing master detail relation dialog box

- 16 Select ItemID as the Field to distinguish this detail from others.

17 Select LineItemPOID as the Field to link the parent form with current form (foreign key).

18 Click OK.

When you click OK, you end up back at the Mapping dialog box.

19 Map the fields as shown in the following table.

Forms and fields	XML elements and complex types	Field to distinguish this detail item from others	Field to link the parent form with current form (Foreign key)
Line Items	ROOT/Items/Item	ItemID	LineItemPOID
Item Name	ItemName		
Quantity	Quantity		
ItemID	ItemId		

20 Click OK to close the Mapping dialog box.

21 Click the Output Mapping button to open the Mapping dialog box.

- The Customized radio button will be selected.
- PO and ROOT will be mapped with Request ID as the primary key.

22 Select PO from the Forms pane (and ROOT from the XML Data Type pane if it is not selected).

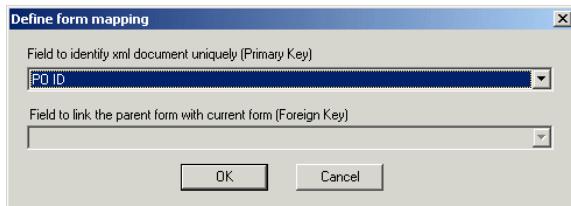
23 Click Unmap.

24 Select PO and ROOT again.

25 Click Map.

The Define form mapping dialog box appears.

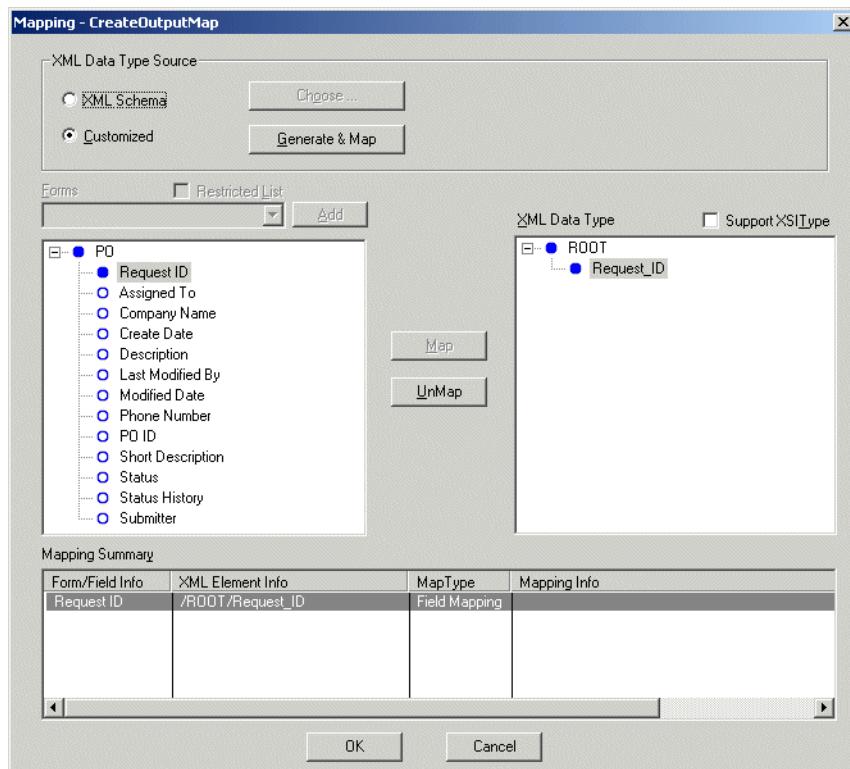
Figure C-21: Define form mapping dialog box for PO output mapping



26 Select PO ID as the primary key, and click OK.

- 27 Select Request ID in the Forms pane and the XML Data Type pane. (It is the only output field.)
- 28 Click Map.

Figure C-22: Output mapping dialog box for PO web service



- 29 Click OK to close the Mapping Dialog box.

Note: Make sure the web service that you created has public permissions. You will see a warning message if your permissions are not set to public. To do this, select the Permissions tab in the Create Web Service dialog box, select Public and click Add.

- 30 Save your web service.

► To map for the GetPurchaseOrder operation

- 1 Select GetPurchaseOrder from the Operations List.
- 2 Click the Input Mapping button.
The Mapping dialog box appears.
- 3 Leave the XML Data Type Source as Customized.
- 4 Click the Generate & Map button.
- 5 Click OK at the warning message that your existing mappings will be replaced.
- 6 Select PO and ROOT and UnMap.
- 7 Select PO and ROOT again and Map.
- 8 Select PO ID as the primary key.

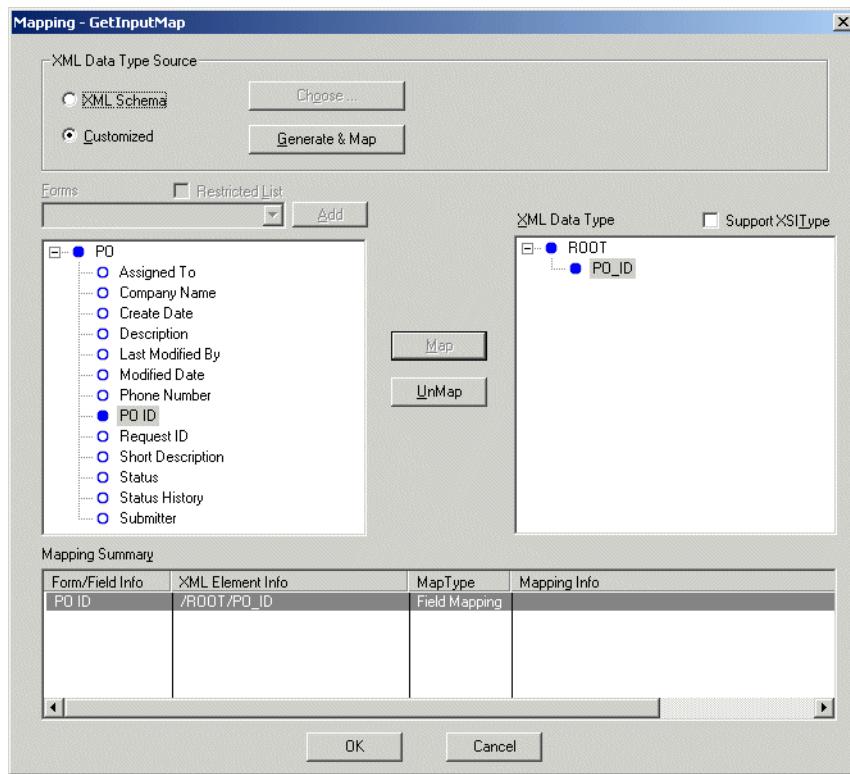
Figure C-23: Define form mapping dialog box for GetPurchaseOrder web service



- 9 Click OK to close the Define form mapping dialog box.
- 10 In the XML Data Type pane, delete all the elements except PO_ID by right clicking on the field name and selecting Cut from the context list.
- 11 Select PO ID in the Forms pane and PO_ID in the XML Data Type pane.
- 12 Click Map.

Your Mapping dialog box should look similar to Figure C-24.

Figure C-24: Input mapping for the GetPurchaseOrder example



- 13 Click OK to close the Mapping dialog box.
- 14 Remove the qualification in the Qualification bar in the Web Services dialog box and replace with the following text:
`'PO_ID'=XPATH(/ROOT/PO_ID)`
 You can either type the entry or use the arrow at the right of the Qualification bar.
 - a Select the PO > PO ID
 - b Click =
 - c Select XPath.
 The Choose XPath dialog box opens.
 - d Select PO_ID and click OK.
- 15 Click the Output Mapping button to open the Mapping dialog box.

- 16 Select the XML Schema option button.
 - 17 Click OK at the warning message that your existing mappings will be replaced.
 - 18 Click the Choose button.
- The Choose start element dialog box appears.
- 19 Select PurchaseOrder as your start element and click OK to close the dialog box.
 - 20 Select PO and ROOT again.
 - 21 Click Map.

The Define form mapping dialog box appears.

Figure C-25: Define form mapping dialog box for GetPurchaseOrder output mapping



- 22 Select PO ID as the Primary Key, and click OK.
- 23 Map the other fields on the PO form to XML elements and complex types as shown in the following table. (However, you would not map Request ID if you were using it as the primary key.)

Forms and fields	XML elements and complex types	Field to identify xml document uniquely
PO	ROOT	PO ID
PO ID	POID	
Description	Description	
Phone Number	PhoneNumber	
Company Name	CompanyName	

24 Select LineItems from the Forms pane and Item from the XML Data Type pane.

25 Click Map.

The Define Form Mapping: Establishing master detail relation form appears.

26 Select ItemID as the Field to distinguish this detail from others.

27 Select LineItemPOID as the Field to link the parent form with current form (Foreign Key).

Figure C-26: Define form mapping: establishing master detail relation dialog box



28 Click OK.

29 Map the fields as shown in the following table.

Forms and fields	XML elements and complex types	Field to distinguish this detail item from others	Field to link the parent form with current form (Foreign key)
Line Items	ROOT/Items/Item	ItemId	POID
Item Name	ItemName		
Quantity	Quantity		
ItemID	ItemId		

30 Click OK to close the Mapping dialog box.

Note: Make sure the web service that you created has public permissions. You will see a warning message if your permissions are not set to public.

31 Save your web service.

► To view your WSDL for the PO web service

- 1 Click the WSDL tab on the Web Service dialog box.

You will see a sample URL for your WSDL file displayed in the field.

Figure C-27: Sample URL displayed in WSDL field



- 2 Complete the URL as appropriate for your configuration, as follows:

- Replace *<midtier_server>* with the name of the web server where the mid tier is running.
- After WSDL, add /public or /protected depending on the web service's permissions.

For example, if the mid tier server is POLYCARP and the web service has *public* permissions, use:

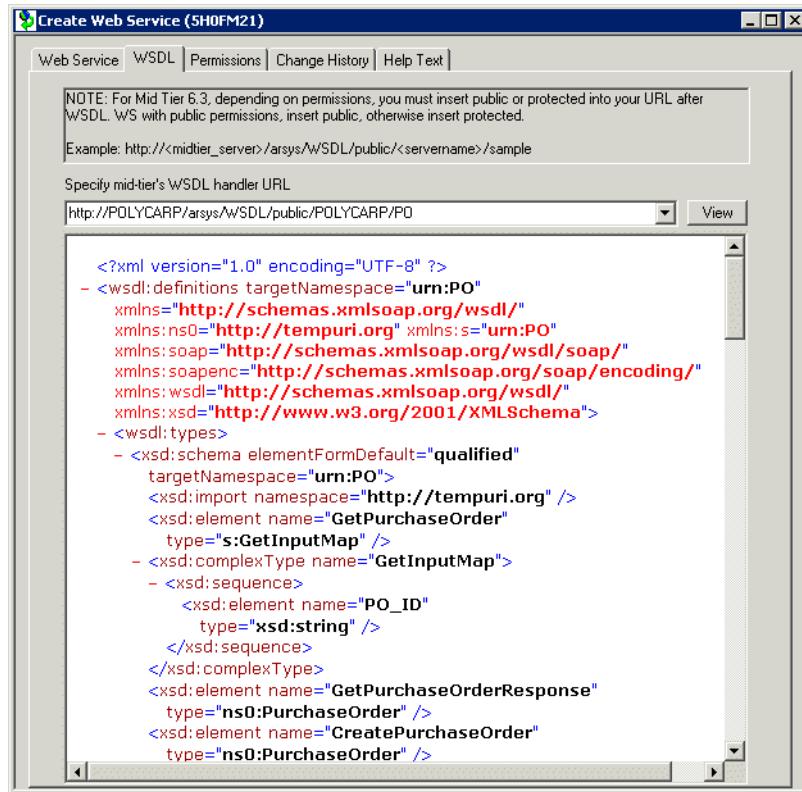
`http://POLYCARP/arsys/WSDL/public/POLYCARP/PO.`

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

`http://POLYCARP/arsys/WSDL/protected/POLYCARP/PO.`

- 3 Click the View button to view your WSDL file in the window.

Figure C-28: WSDL file for CreatePurchaseOrder web service



- 4 Enter your name and password at the prompt, if necessary, and click Login. Your WSDL file is displayed in the View field window as shown.

When you invoke the CreatePurchaseOrder Web Service, the Request ID will be returned if the Web Service has been successful.

Example 4: consuming a complex document

In this example you will access the PO web service you created in Example 3, and get a purchase order and a line item record with the use of the Set Fields Web Service filter.

Note: Make sure the web service that you created has public permissions.

The process for this example of consuming a complex document is as follows:

- Step 1** Create the forms. (see page 441)
- Step 2** Create the input mappings. (see page 444)
- Step 3** Create the output mappings. (see page 446)
- Step 4** Consume the web service. (see page 448)
- Step 5** View and debug the web service. (see page 448)

For information about setting your environment to consume a web service created on the same AR System server, see “Consuming a web service published on the same AR System server” on page 109.

► To create your forms

- 1 Create two new forms, following the procedure in Example 3, step 1 to step 26, but name your forms PO Client and Line Items Client.

Your LineItemsClient form should look similar to the one shown in Figure C-29.

Figure C-29: Line items client form

The screenshot shows the 'Modify Form - Line Items Client (SHOFM21)' window. The title bar includes the view 'Default Admin View [Windows]' and the form name 'Line Items Client'. The main area is a grid where fields are mapped. The fields visible include:

- Request ID (text box)
- Submitter (\$USER\$ placeholder)
- Create Date (date picker)
- Assigned To (text box)
- Last Modified By (text box)
- Modified Date (date picker)
- Status (radio buttons: New, Assigned, Fixed, Rejected, Closed)
- LineItemPOID (text box)
- ItemID (text box)
- ItemName (text box)
- Quantity (text box)
- Short Description (\$TIMESTAMP\$ placeholder)

Below the grid, there are buttons for 'Field Properties' and 'Refresh Field'. At the bottom, it says 'View Size : [W] 708 [H] 318'.

Your PO Client form should look similar to the one shown in Figure C-30.

Figure C-30: PO client form

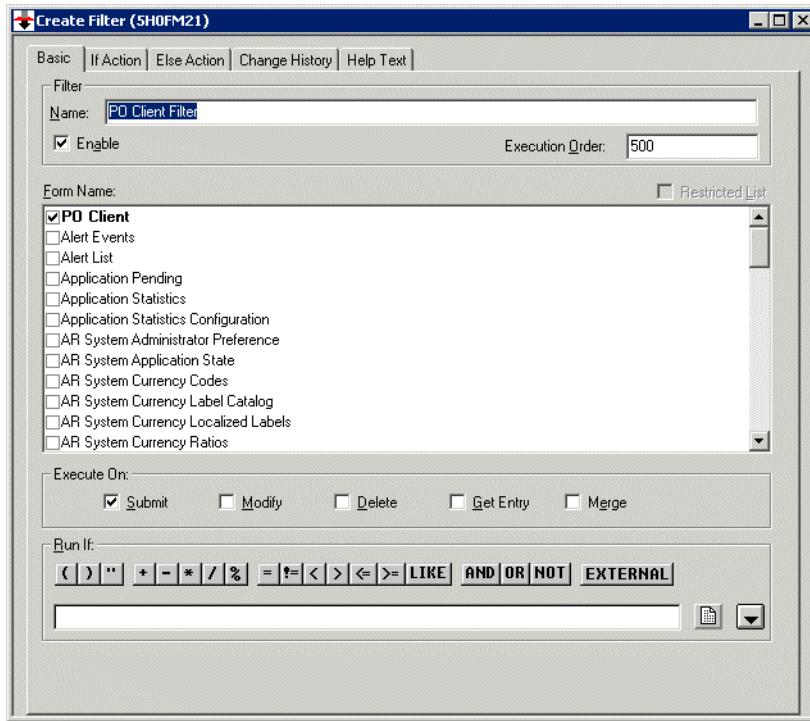
The screenshot shows the 'Modify Form - PO (5H0FM21)' window. At the top, it says 'View: Default Admin View [Windows]' and 'Form: PO'. There is a 'Find Field' dropdown set to 'Table Field(Table Field)', a 'Field Properties' button, and a 'Refresh Field' button. A link 'Choose a page in web view:' is also present. The main area contains several input fields: 'Request ID' (text box), 'Submitter' (text box containing '\$USER\$'), 'Create Date' (text box with a calendar icon), 'Assigned To' (text box), 'Last Modified By' (text box), 'Modified Date' (text box with a calendar icon), 'Status' (radio buttons for New, Assigned, Fixed, Rejected, Closed), and 'Short Description' (text box containing '\$TIMESTAMP\$'). To the right, there are four more input fields: 'PO ID' (text box), 'Company Name' (text box), 'Description' (text box), and 'Phone Number' (text box). Below these is a table grid with three columns: 'ItemID', 'ItemName', and 'Quantity'. The table has a single row with empty cells. At the bottom left, it says 'View Size : [W] 789 [H] 369'.

Tip: Choose File > Save Form As to save the forms with new names.

In the server window:

- 2 Choose File > New Server Object.
- 3 Click Filter and OK to display the Create Filter dialog box, Basic tab.
- 4 Enter a name for the filter, PO Client Filter.
- 5 Check the PO Client form to use as the base form.
- 6 Check Execute On Submit.

Figure C-31: Create filter dialog box for PO client filter



- 7 Click the If Action tab.
- 8 Select Set Fields from the New Action field list.
- 9 In the Server name field, select the name of the server on which the Web Service filter plug-in is installed.
- 10 Select WEB SERVICE in the Read Value for Field From field.
- 11 Type the path to your WSDL file in the Choose WSDL field, or browse to select it.

Your URL might look like this:

`http://POLYCARP/arsys/WSDL/public/POLYCARP/PO`

You can also type the path to a local WSDL file; for example:

`C:\Temp\PO.wsdl`

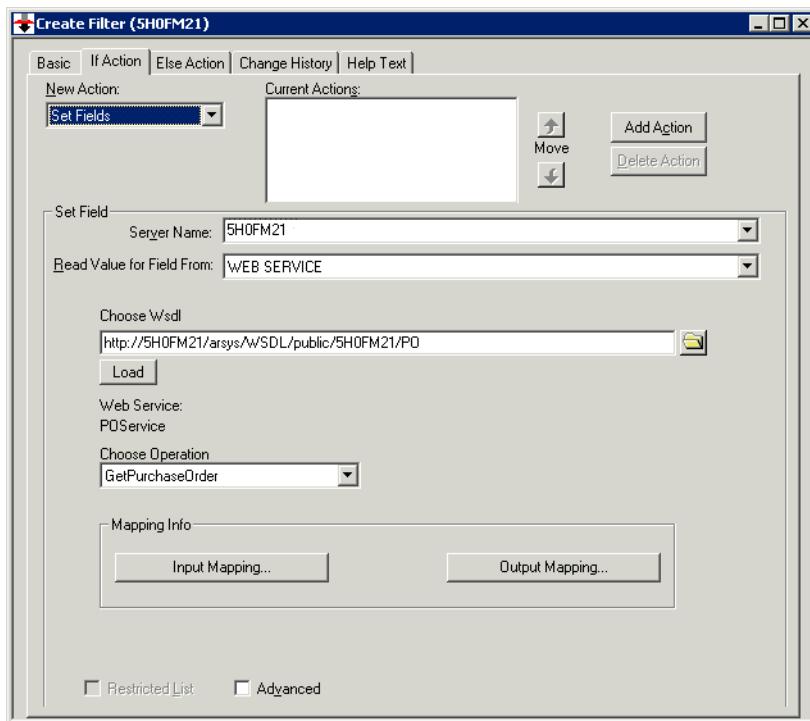
12 Click Load.

The name of the web service (PO) is displayed on the dialog box under the Web Service heading.

The available operations for the PO web service are automatically listed.

13 Choose GetPurchaseOrder from the Choose Operations drop-down list.

Figure C-32: If action tab of the create filter dialog box

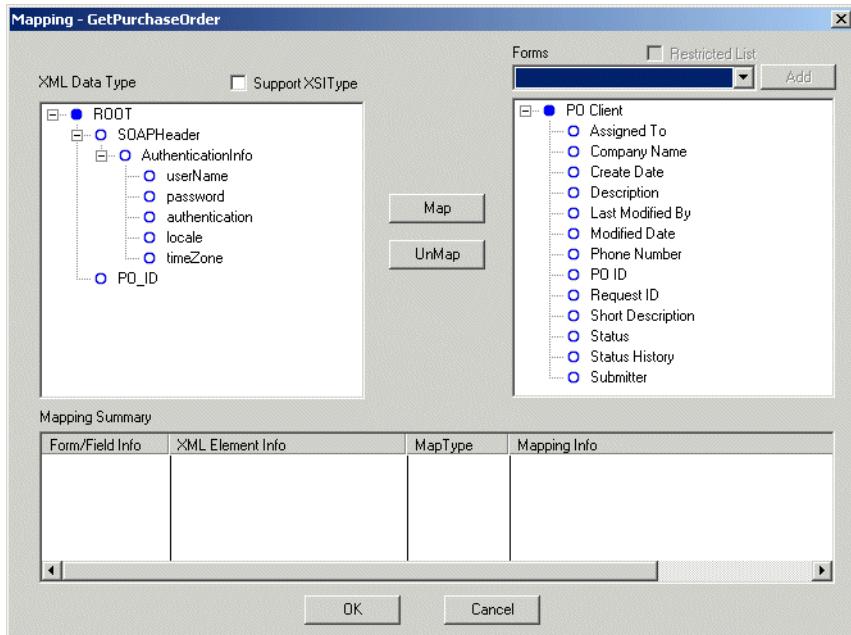


► **To create input mappings**

- 1 Click the Input Mapping button to display the Mapping dialog box.

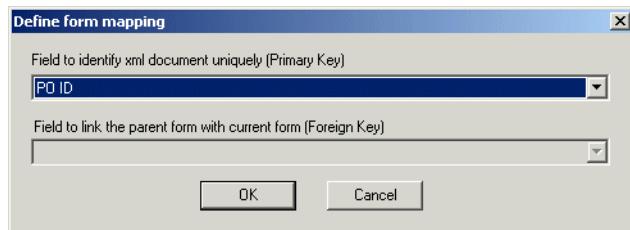
The ROOT element of the XML schema and the PO Client form are automatically mapped as shown in the following figure.

Figure C-33: Input mapping for POClient consuming web service



- 2 Select ROOT and PO Client and click Unmap.
- 3 Select ROOT and PO Client again and click Map (the circles next to these two items should become solid).
- 4 Select PO ID as the primary key in the Define form mapping dialog box.

Figure C-34: Define form mapping dialog box for GetPurchaseOrder



- 5 Click OK.
- 6 In the Mapping dialog box, select PO_ID and PO ID and click Map.
- 7 Click OK.

► To create output mappings

- 1 In the Create Filter dialog box, click Output Mapping.
- 2 Select ROOT and PO Client and click Unmap.
- 3 Select ROOT and PO Client again and click Map (the circles next to these two items should become solid).
- 4 Select PO ID as the primary key in the Define form mapping dialog box.
- 5 Map the following elements to fields.

XML elements and Complex types	Fields
ROOT	POClient
POID	PO ID
CompanyName	Company
Description	Description
PhoneNumber	Phone Number

- 6 Add the Line Items Client form to the Forms pane by clicking the Forms arrow, selecting the Line Items Client form and clicking Add.
- 7 Select Item in the XML Data Type pane and Line Items Client in the Forms pane.
- 8 Click Map.

The Define Form mapping: Establishing master detail relation form opens.

- 9 Select ItemID for the Field to distinguish this detail item from others.
- 10 Select LineItemPOID as the foreign key.
- 11 Click OK.

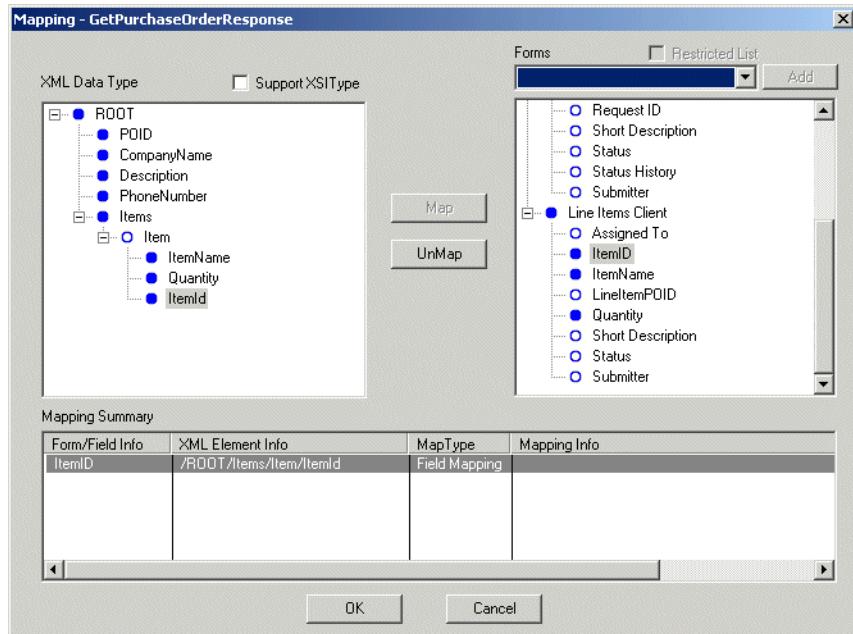
- 12** Map the following elements and fields:

XML elements and complex types

Item	Line Items Client
ItemName	ItemName
Quantity	Quantity
ItemId	ItemID

Your mapping should look similar to Figure C-35.

Figure C-35: Output mapping for GetPurchaseOrder consumption



- 13** Click OK to close the Mapping dialog box.
- 14** In the Create Filter dialog box, click Add Action to add the filter to the Current Actions pane.
- 15** Save the filter.

► **To consume the web service in an AR system client**

- 1 Open an BMC Remedy User client and log in.
- 2 Open a new PO form (created in “Example 3: publishing a complex document” on page 423).
- 3 Enter data in the required fields, including a PO ID to a value of 111.
- 4 Save the record.
- 5 Open a new Line Items form (created in “Example 3: publishing a complex document” on page 423).
- 6 Enter data in the required fields, including a LineItemPOID to a value of 111.
- 7 Save the record.
- 8 Open a new PO Client form.
- 9 Enter a value in the PO ID field of 111.
- 10 Save the form.

You created the POClient filter to trigger On Submit.

- 11 Open the request you created in step 9 in the POClient form.

If the GetPurchaseOrder Web Service was successful, the request is submitted and contains the same values as those of the record in the PO form whose PO ID was set to the value 111.

The values corresponding to the request you created in step 6 for the LineItems form will appear also in the LineItemsClient form.

► **To view and debug your web service**

- 1 Stop the AR System server.
- 2 Open the armonitor.cfg file.

The default location is

C:\Program Files\AR System\Conf

- 3 Comment out the line similar to:

```
"C:\Program Files\AR System\arplugin.exe" -i "C:\Program  
Files\AR System\" -m
```

- 4 Copy the line and run it in the Command Line.
- 5 Make sure the message Loaded Web Services plugin properly is displayed.

D

Sample exercise: using OLE automation

The following exercise describes how to create a form and an active link to open Microsoft Word, spell check text in a field in that form, and return the corrected text back to the field.

The following topics are provided:

- Procedure 1—Creating the form (page 450)
- Procedure 2—Defining the active link (page 451)
- Procedure 3—Defining action 1 (page 452)
- Procedure 4—Defining action 2 (page 454)
- Procedure 5—Defining action 3 (page 456)
- Procedure 6—Defining action 4 (page 458)
- Procedure 7—Defining action 5 (page 459)
- Procedure 8—Testing the active link (page 462)

Note: This example is only a partial solution that does not cover all aspects of this problem. For a more detailed example that uses a different approach, see the Sample: SpellCheck active link in the ClassCentral sample application.

To complete the exercise, 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.

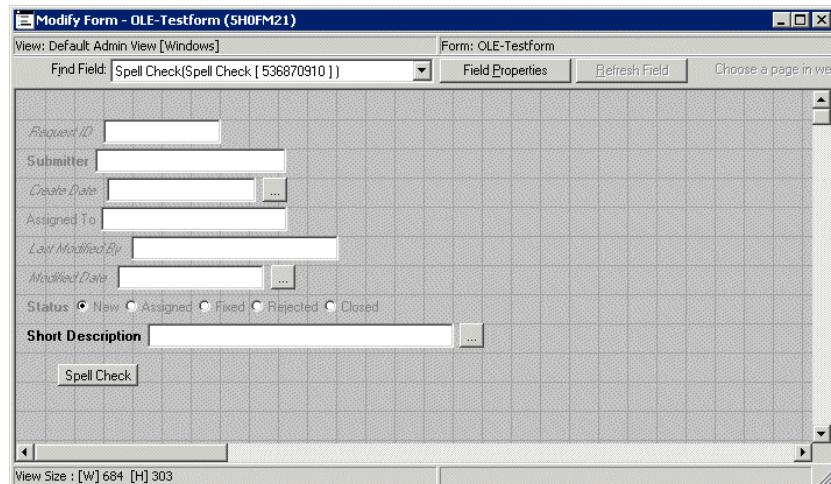
Procedure 1—Creating the form

Use the following procedure to create the form.

► To create the form

- 1 In BMC Remedy Administrator, select a server, and choose File > New Server Object.
- 2 Select Regular Form, and click OK to create a new form.
- 3 Create a new button located under the Short Description field, with the following properties:
 - Button Label: Spell check
 - Button Name: Spell check
- 4 Using the Permission tabs, set permissions to Public for the Short Description field and the Spell Check button.
- 5 Hide the other fields on the form and move the position of the Short Description field and Spell Check button, if desired.
- 6 Name the form OLE-Testform, and save it. (Your form will be similar to Figure D-1.)

Figure D-1: OLE -Testform



- 7 See the following procedures to create the active link and to set up the active link actions for the Spell Check button.

Procedure 2—Defining the active link

Use the following procedure to define the active link.

► To define the active link

1 In BMC Remedy Administrator, select a server, and choose File > New Server Object.

2 Select Active Link, and click OK to create a new active link.

3 Enter OLE-APP-CheckSpel1 in the Name field.

This is the name of the OLE automation active link.

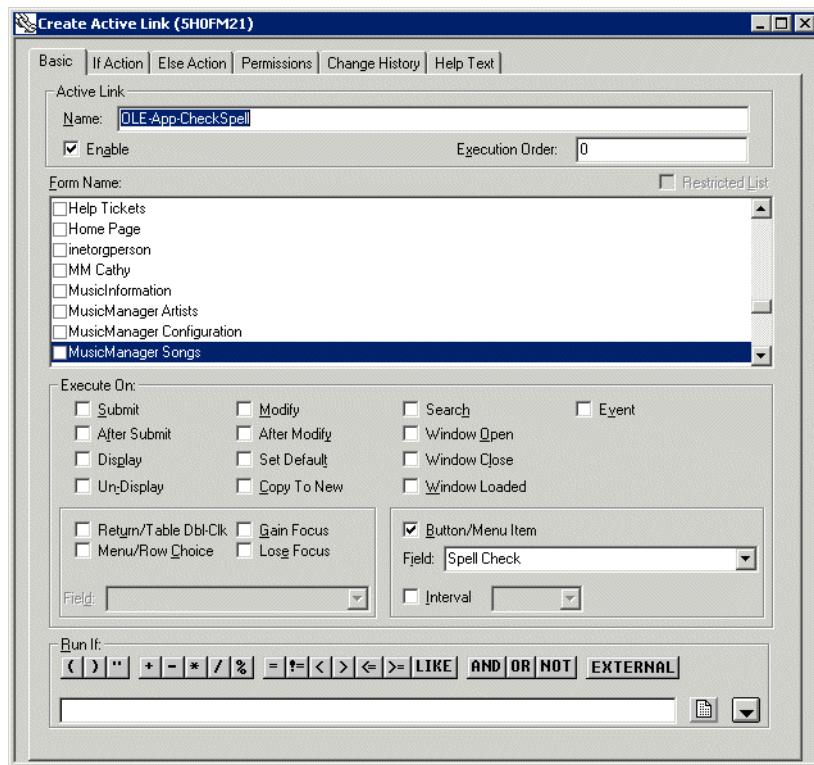
4 Select OLE-Testform, which is the form that you created by using the procedure “To create the form” on page 450.

5 Select the Button/Menu Item check box.

6 Select the Spell Check button from the Field list, as shown in Figure D-2 on page 452.

This specifies that the active link is invoked when a user clicks the Spell Check button.

Figure D-2: Create active link dialog box



Procedure 3—Defining action 1

The action described in this procedure makes Microsoft Word visible. When you create this action for the active link, you will construct the following objects:

```
_Document Object::Application*Application( )
_Application Object::void Visible( 1 )
```

► To define action 1

- 1 In the If Action tab of the Create Active Link dialog box, select OLE Automation from the New Action list.
- 2 In the Automation Servers list, expand OLE Local Servers.
- 3 Double-click the Microsoft Word Document local server.

The type library information appears in the Type Library Information list.

-
- 4 Expand Objects in the Type Library Information list.

The list of objects appear for the Microsoft Word Document local server.

- 5 Expand _Document Object in the Type Library Information list, and expand Methods below it.

- 6 Select the Application* Application() method, and click Add Method.

Application appears in the Parameter List.

- 7 Select Application in the Parameter List.

By selecting Application in the Parameter List, you are specifying that the next method that you select will be placed below it. This is a new OLE automation action. Nevertheless, BMC Remedy User treats this as a continuation of the previous stream of OLE method calls. The division of the stream into multiple actions is arbitrary.

- 8 Expand _Application Object in the Type Library Information list, and expand Methods below it.

- 9 Select the void Visible(boolean rhs) method, and click Add Method.

The Visible method appears in the Parameter List below Application.

- 10 Set parameters for the void Visible(boolean rhs) method by using the Parameter List.

a Expand Visible.

b Expand Parameters.

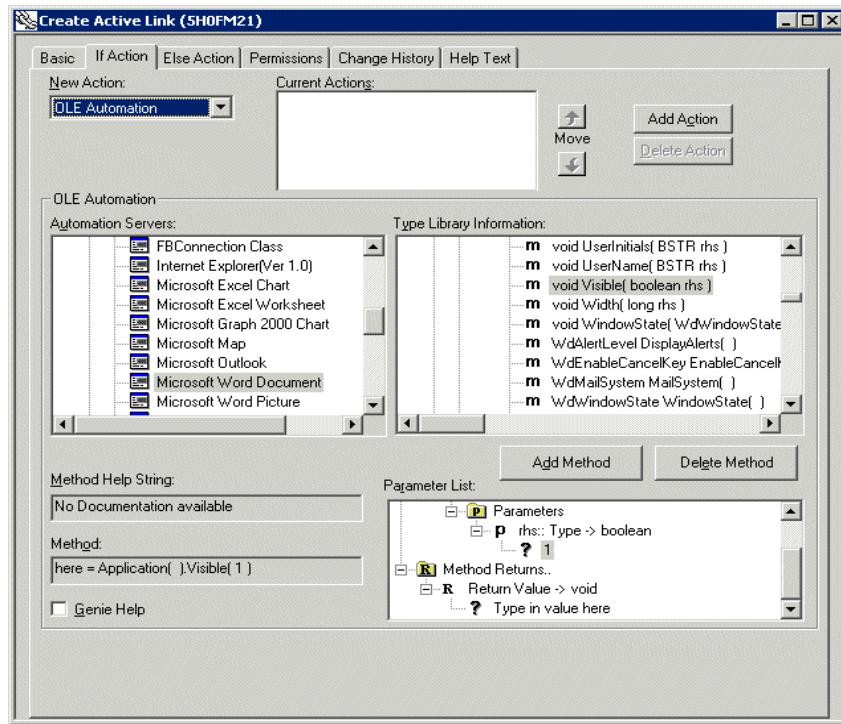
c Expand rhs:: Type->boolean

d Select ?Type in value here and enter 1.

This sets the Boolean parameter to 1 (TRUE).

Note: To add a field or keyword as the parameter value, right-click ?Type in value here to open a list of form fields or keywords. Select the appropriate choice from the list.

The Parameter List appears as shown in Figure D-3 on page 454.

Figure D-3: Parameter list

- 11 Click Add Action and save the action.

Procedure 4—Defining action 2

The action described in this procedure transfers the contents of the Short Description field to Microsoft Word. When you create this action for the active link, you will construct the following objects:

```
Document Object::Sentences()
Sentences Object::Range*First()
Range Object::void InsertAfter (BSTR Text)
Set text to $Short Description$
```

► To define action 2

- 1 Select OLE Automation from the Current Actions list, and select OLE Automation from the New Action list.

By selecting the first action, the next action that you create is placed below it. This is a new OLE Automation action; yet, BMC Remedy User treats this as a continuation of the previous stream of OLE method calls. The division of the stream into multiple actions is arbitrary.

- 2 Expand _Document Object in the Type Library Information list and expand Methods below it.

- 3 Select the Sentences* Sentences() method, and click Add Method.

Sentences appears in the Parameter List.

- 4 Select Sentences in the Parameter List.

By selecting Sentences in the Parameter List, you are specifying that the next method that you select will be placed below it.

- 5 Expand Sentences Object in the Type Library Information list, and expand Methods below it.

- 6 Select the Range*First() method, and click Add Method.

First appears in the Parameter List below Sentences.

- 7 Select First in the Parameter List.

By selecting First in the Parameter List, you are specifying that the next method that you select will be placed below it.

- 8 Expand Range Object in the Type Library Information list, and expand Methods below it.

- 9 Select the void InsertAfter (BSTR Text) method, and click Add Method.

InsertAfter appears in the Parameter List below First.

- 10 Set parameters for InsertAfter by using the Parameter List.

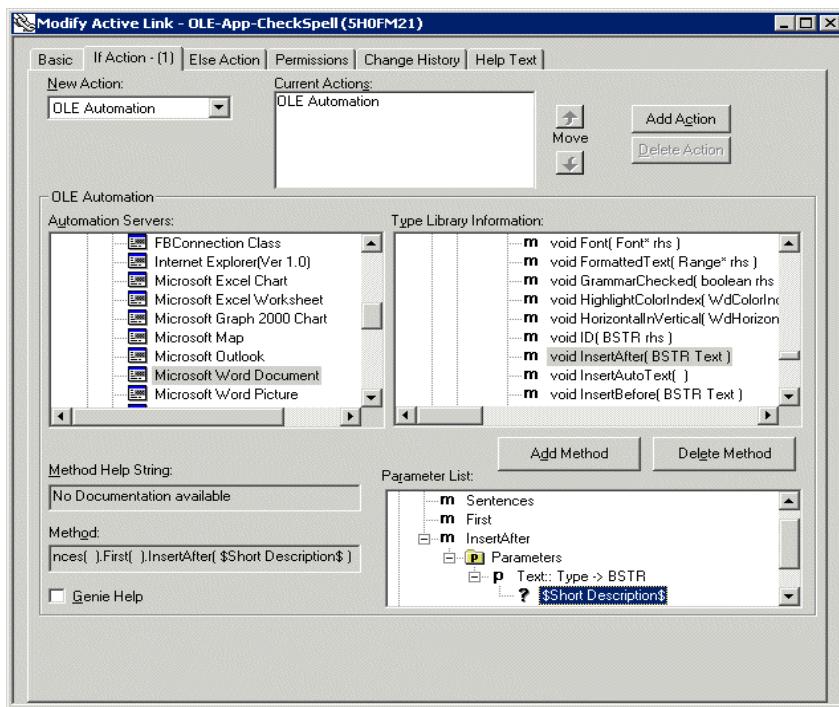
- a Expand InsertAfter, and expand Parameters below it.

- b Expand the Text>Type -> BSTR parameter.

- c Select ?Type in Value here, right-click, and select Fields > Short Description.

The values in the Method field and in the Parameter List are shown in Figure D-4 on page 456.

Figure D-4: Action two values



- Click Add Action, and save the action.

Procedure 5—Defining action 3

The action described in this procedure spell checks the contents of the Short Description field. When you create this action for the active link, you will construct the following object:

```
_Document Object:void CheckSpelling( )
```

► To define action 3

- Select the last OLE Automation from the Current Actions list, and select OLE Automation from the New Action list.

By selecting the last action in the list, the next action that you create is placed below it.

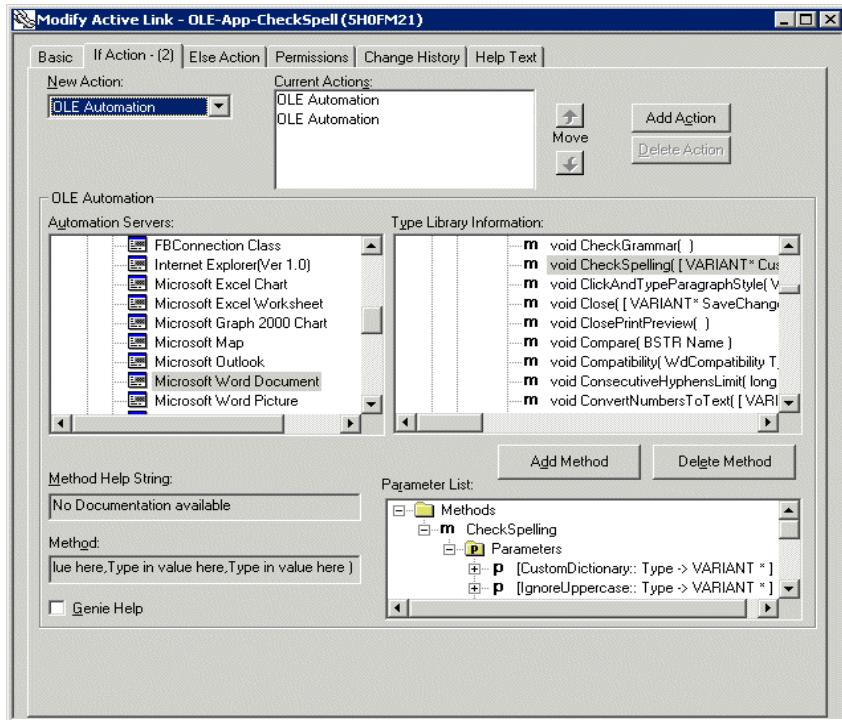
- 2 Expand _Document Object in the Type Library Information list, and expand Methods below it.
- 3 Select the void CheckSpelling method, and click Add Method.

CheckSpelling appears in the Parameter List. you do not need to change the parameters for it because they are optional.

Note: Braces [] around the parameters in the Parameters List indicate that the parameters are optional.

If you expand the CheckSpelling method and Parameters, the following values are shown in the following figure.

Figure D-5: Action three values



- 4 Click Add Action, and save the action.

Procedure 6—Defining action 4

The action described in this procedure selects the entire contents of the text in Microsoft Word in preparation for sending it back to the Short Description field. When you create this action for the active link, you will construct the following objects:

```
_Document Object::Application*Application( )  
_Application Object::Selection* Selection( )  
Selection Object::void WholeStory( )
```

► To define action 4

- 1 Select the last OLE Automation in the Current Actions list, and select OLE Automation from the New Action list.

By selecting the last action in the list, the next action that you create is placed below it. This is a new OLE automation action. Nevertheless, BMC Remedy User treats this as a continuation of the previous stream of OLE method calls. The division of the stream into multiple actions is arbitrary.

- 2 Expand _Document Object in the Type Library Information list, and expand Methods below it.

- 3 Select the Application* Application() method, and click Add Method.

Application appears in the Parameter List.

- 4 Select Application in the Parameter List.

By selecting Application in the Parameter List, you are specifying that the next method that you select will be placed below it.

- 5 Expand _Application Object in the Type Library Information list, and expand Methods below it.

- 6 Select the Selection* Selection() method, and click Add Method.

Selection appears in the Parameter List below Application.

- 7 Select Selection in the Parameter List.

By selecting Selection in the Parameter List, you are specifying that the next method that you select will be placed below it.

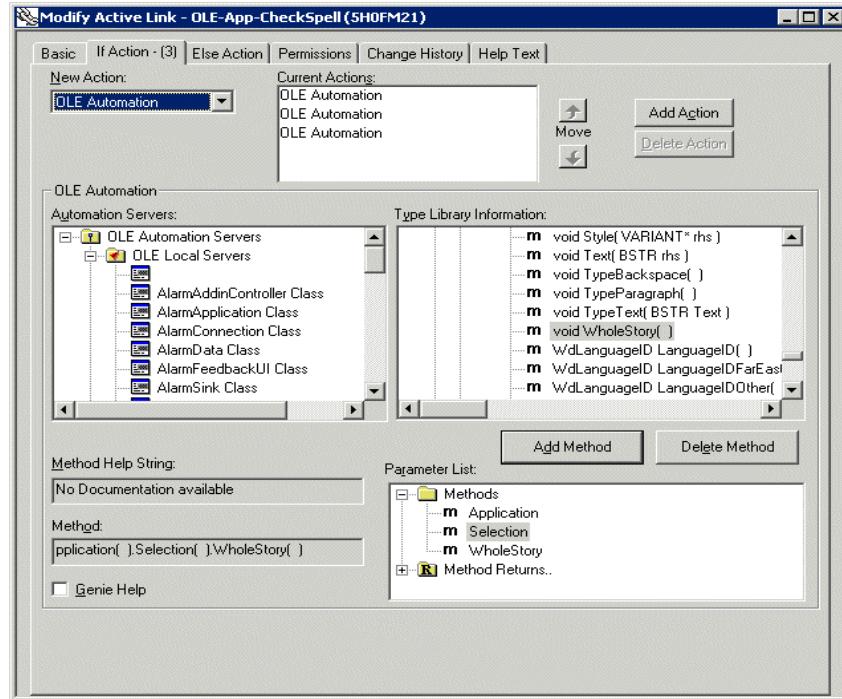
- 8 Expand Selection Object in the Type Library Information list, and expand Methods below it.

- 9 Select the void WholeStory() method, and click Add Method.

WholeStory appears in the Parameter List below Selection.

The values in the Method field are shown in Figure D-6 on page 459.

Figure D-6: Action four values



- 10 Click Add Action and save the action.

Procedure 7—Defining action 5

The action described in this procedure sends the selected text back to the Short Description field. When you create this action for the active link, you will construct the following objects:

```
_Document Object::Application*Application( )
_Application Object::Selection* Selection( )
Selection Object::BSTR Text( )
Set Text to $Short Description$
```

► To define action 5

- 1 Select the last OLE Automation in the Current Actions list, and select OLE Automation from the New Action list.

By selecting the last action in the list, the next action that you create is placed below it. This is a new OLE Automation action. Nevertheless, BMC Remedy User treats this as a continuation of the previous stream of OLE method calls. The division of the stream into multiple actions is arbitrary.

- 2 Expand _Document Object in the Type Library Information list, and expand Methods below it.

- 3 Select the Application* Application() method, and click Add Method.

Application appears in the Parameter List.

- 4 Select Application in the Parameter List.

By selecting Application in the Parameter List, you are specifying that the next method that you select will be placed below it.

- 5 Expand _Application Object in the Type Library Information list, and expand Methods below it.

- 6 Select the Selection* Selection() method, and click Add Method.

Selection appears below Application in the Parameter List.

- 7 Select Selection in the Parameter List.

By selecting Selection in the Parameter List, you are specifying that the next method that you select will be placed below it.

- 8 Expand Selection Object in the Type Library Information list, and expand methods below it.

- 9 Select the BSTR Text() method, and click Add Method.

Text appears in the Parameter List below Selection.

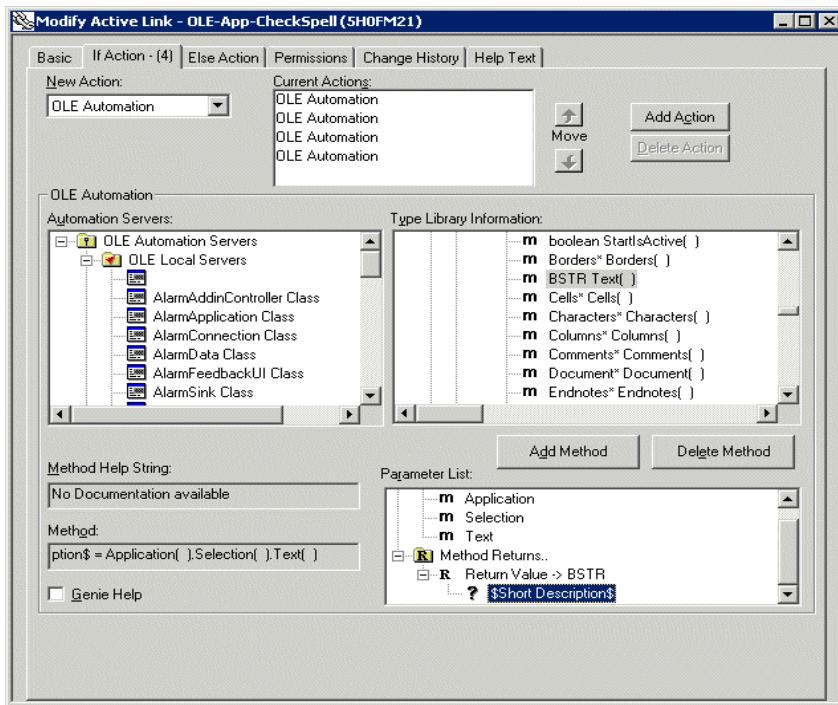
- 10 Set the return values by using the Parameter List.

- a Expand Method Returns.

- b Expand Return Value -> BSTR.

- c Select ?Type in Value here, right-click, and select Fields > Short Description.

The values in the Method field, and in the Parameter List are shown in Figure D-7 on page 461.

Figure D-7: Action five values

11 Click Add Action, and save the action.

Your active link is now complete and you are ready to test it (by using Microsoft Word) as described in “To test the active link.”

Make sure that you distribute the local OLE automation server to users’ machines so that the active link executes without error. Also, make sure that the local OLE automation server is registered on users’ machines by using the Regsvr32 utility (this should occur during the installation of Microsoft Office). For more information about registering local OLE automation server controls, see your Microsoft OLE automation documentation.

Note: You can modify and delete active link actions by selecting the action from the Current Actions list, and clicking Modify Action or Delete Action.

Procedure 8—Testing the active link

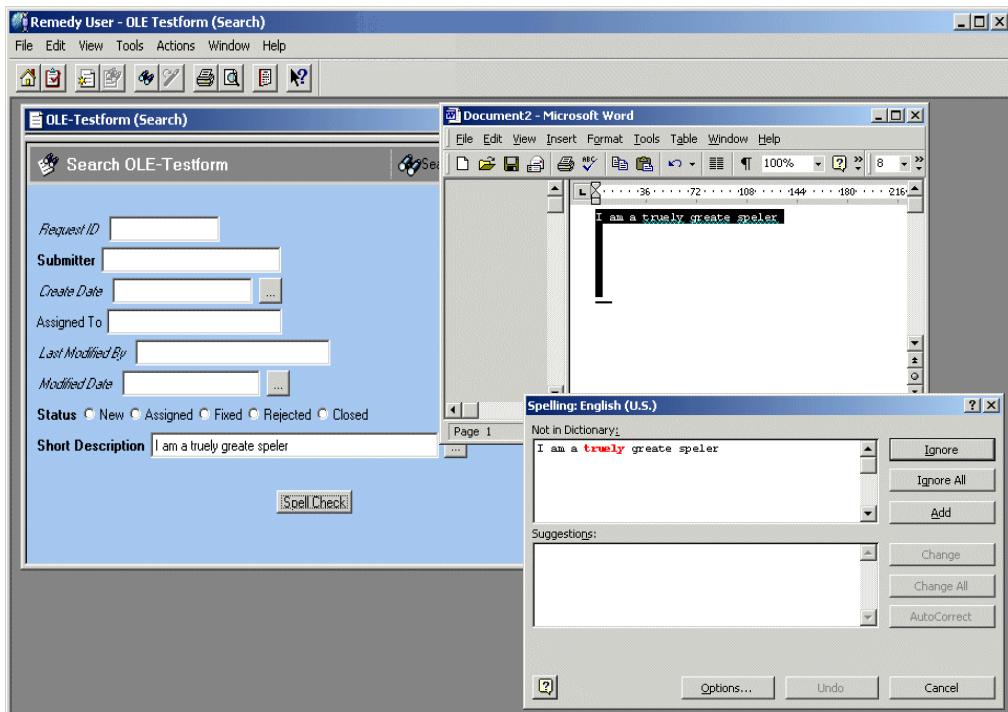
Use the following procedure to test the active link.

► To test the active link

- 1 Open BMC Remedy User.
- 2 Choose File > Open.
- 3 In the Open dialog box, select the OLE-Testform that you created in the sample exercise. Click New to open a new form.
- 4 Enter misspelled text in the Short Description field.
- 5 Click Spell Check.

The Search OLE-Testform appears as shown in the following figure, which enables the user to correct the spelling of the text in the Short Description field by using Microsoft Word.

Figure D-8: Spell check form



- 6 Correct the spelling by using the Microsoft Word spell checking feature.
- 7 Exit Microsoft Word, and return to BMC Remedy User.
- 8 View the changed text.

E 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 466)
- Attaching fields to represent `inetorgperson` data (page 469)
- Defining a filter to generate a DN (page 472)

Note: You must have your ARDBC plug-in installed and correctly configured before you can create a vendor to use the plug-in. See “Creating plug-ins” on page 118 and “Configuring the ARDBC LDAP plug-in” on page 141.

Creating the inetorgperson vendor form

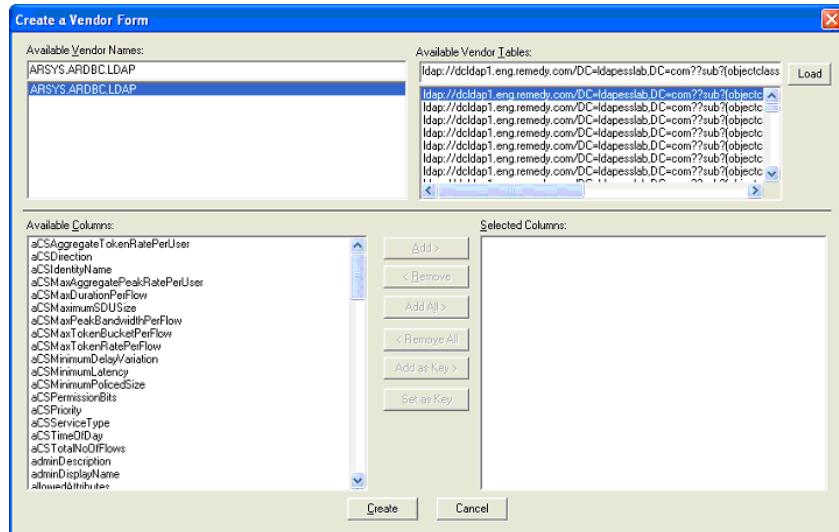
The `inetorgperson` object class is often present by default on some LDAP directory services, such as iPlanet and OpenLDAP. To use the example in this appendix if your LDAP service does not contain the `inetorgperson` objectclass, replace the `objectclass` filter in the `inetorgperson` vendor form. The 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 `<ar_install>\AR\System\Plugins\ARDBC` folder. You can import this definition file using BMC Remedy Administrator.

► To create a vendor form using the `inetorgperson` objectclass

- 1 Start BMC Remedy Administrator and log in to the AR System server.
- 2 Select a server to administer.
- 3 Choose File > New Server Object.
- 4 Select Vendor Form from the list of New Server Objects.
- 5 Select a vendor name, for example, ARSYS.ARDBC.LDAP, in the Available Vendor Names field.
- 6 Select the LDAP URL in the Available Vendor Tables field that corresponds to the `inetorgperson` object class:

```
ldap://<ldap_directory_service_host>/  
o=remedy.com??sub?(objectclass=inetOrgPerson)
```

Note: If your LDAP server does not contain the `inetorgperson` objectclass, select a comparable objectclass, for example, `person`.

Figure E-1: Create a vendor form dialog box

7 Select one or more available columns.

8 Click Create to create the form.

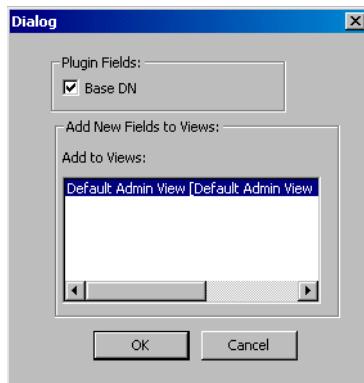
A form with a Request ID field appears. Adjust fields as needed.

Figure E-2: Create form window

- 9 Optionally, you can add a BaseDN field, in which the user can specify a base DN to act as the starting point for queries.
- a Select Form > Plug-in Fields.

The Plug-in Fields dialog box appears.

Figure E-3: Plug-in fields dialog box



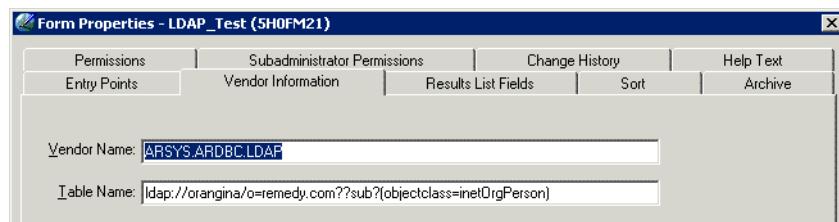
- b In the Plug-in Fields group box, select the Base DN check box.
- c In the Add to Views list, select the views to which you want to add the Base DN field and click OK.

The Base DN field is added to your vendor form.

- 10 Choose File > Save Form As to display the Save Vendor Form As dialog box.
- 11 Enter the form name in the Form Name field and click OK to save the form.

You can modify the LDAP search URL at any time. In the Form Properties dialog box, you can also specify that the form make use of the ARDBC LDAP plug-in. Other ARDBC plug-ins will 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 E-4: Form properties window showing the LDAP search URL



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 will not return the table of results if the base distinguished name used to search for entries is not specific enough.

12 To add or delete fields from a vendor form:

- To add a field to the vendor form, choose Form > Create a New > Field From <*external_name*> from the menu. Select the field you want to add from the Add Field dialog box, and then click OK.
- To delete a field from the vendor form, click on a field and choose Edit > Delete. Deleted fields are returned to the Available Columns list box for later access if needed. This only deletes the AR System field. It does not remove the column from the database table.

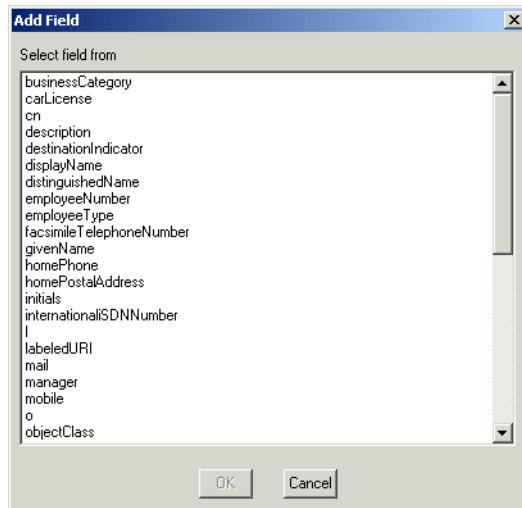
Attaching fields to represent inetorgperson data

After you create a vendor form, you populate the form with fields that will 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 Administrator, open the vendor form you created earlier.
- 2 Right-click in the form and choose Field from `ldap://<ldap_directory_service_host>/o=remedy.com??sub?(objectclass=inetorgperson)` from the menu that appears.

The Add Field dialog box appears.

Figure E-5: Add field dialog box

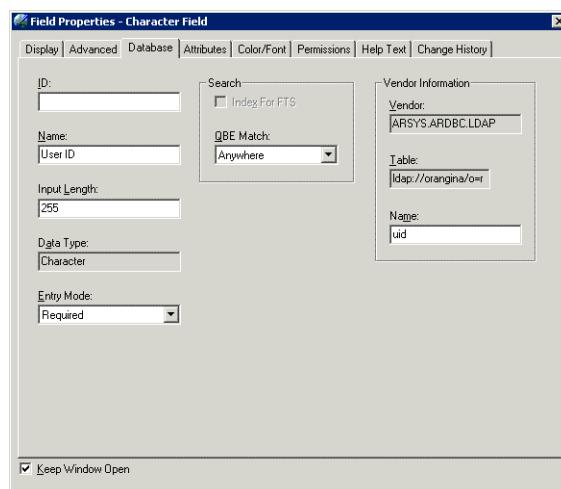
- 3** Select the field to add.

The field appears on your form.

- 4** Position the field as required.

- 5** Double-click the field to open the Field Properties dialog box.

- 6** Click the Database tab to display the database and vendor properties associated with the field.

Figure E-6: Field properties window showing the attribute name

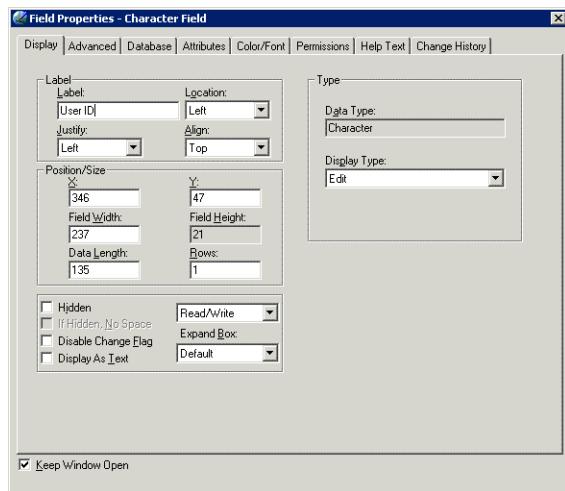
- 7 In the Name field in the Vendor Information box, enter the attribute name, uid, in the Vendor Information Name field.
- 8 Choose File > Save Form to save the field properties.

► **Alternative method of adding a field to represent the uid (User ID) attribute**

- 1 Open the form to which you want to add a field.
- 2 Choose Form > Create a New > Character Field to add a character field to the form.
- 3 With the new Character Field selected, choose Form > Field Properties to display the new field's Field Properties.

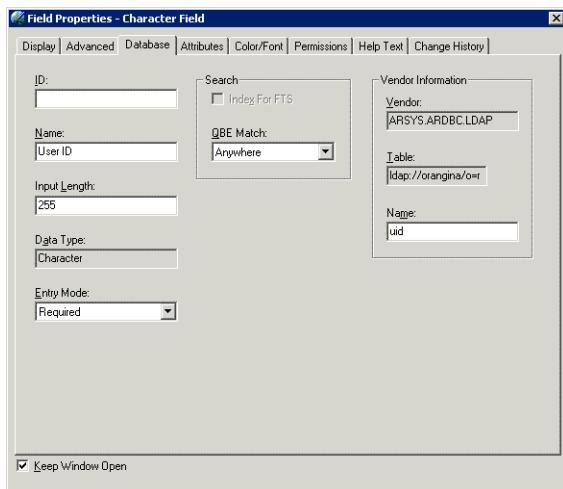
Change the Label to User ID.

Figure E-7: Field properties window—Display tab



- 4 Click the Database tab to display the database and vendor properties associated with the field.

Figure E-8: Field properties window—Database tab



- 5 Choose Required as the Entry Mode.
- 6 In the Name field in the Vendor Information box, enter the attribute name, uid, in the Vendor Information Name field.
- 7 Position the field as required.
- 8 Choose File > Save Form to save the field properties.

Defining a filter to generate a DN

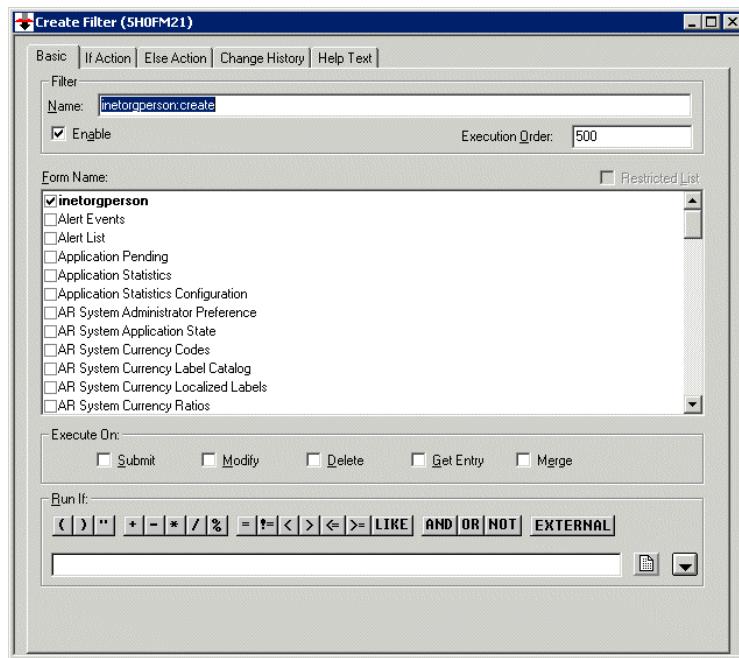
In the `inetorgperson` example, an object's distinguished name looks something like this:

`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.

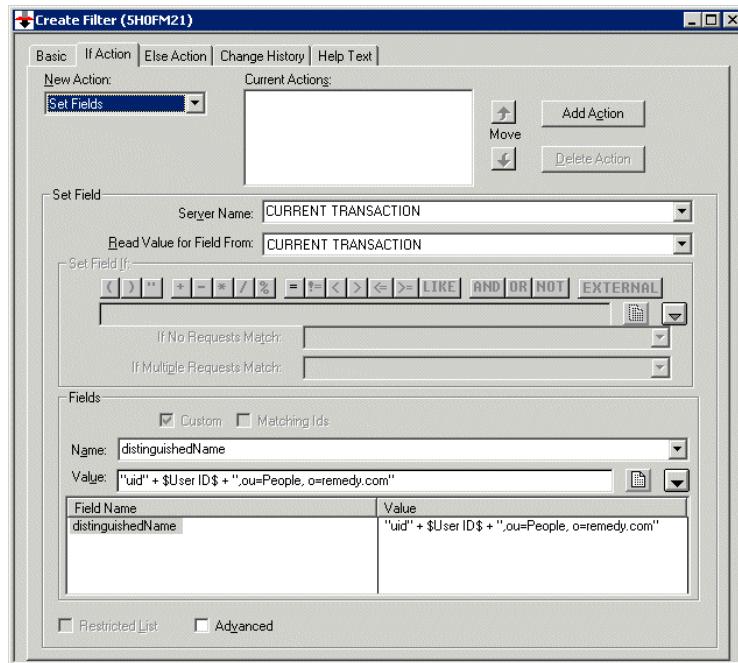
► **To define an AR System filter to construct the distinguished name using the `inetorgperson` example**

- 1 Choose File > New Server Object.
- 2 Select Filter from the list of New Server Objects.
- 3 Enter `inetorgperson:create` as the filter's name.
- 4 Select `inetorgperson` under the Form Name list.

Figure E-9: Create filter form

- 5 Select Submit as an Execute On condition.
- 6 Click the If Action tab.

Figure E-10: Creating a set fields filter action



- 7 Select Set Fields for the New Action.
- 8 Select Distinguished Name in the Name field.
- 9 Enter the following into the Value field:
"uid=" + \$User ID\$ + ", ou=People, o=remedy.com"
- 10 Click Add Action.
- 11 Choose File > Save Filter.

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:

Table E-1: Summary of fields

Field	Field properties
Distinguished Name	Entry Mode: Optional Read/Write Default Value: none Vendor Information Name: dn
Object Class	Entry Mode: Required Read Only Default Value: top, person, organizationalPerson, inetorgperson Vendor Information Name: objectclass[* ,]
Last Name	Entry Mode: Required Read/Write Default Value: none Vendor Information Name: sn
Common Name	Entry Mode: Required Read/Write Default Value: none Vendor Information Name: cn
Organization Unit	Entry Mode: Required Read/Write Default Value: People Vendor Information Name: ou[* ,]

In this example, the form looks like the following figure.

Figure E-11: The inetorgperson form

The screenshot shows a Windows application window titled "inetorgperson (Search)". The window has a toolbar with icons for search and advanced options. The main area is titled "inetorgperson". It contains seven input fields with labels and asterisks indicating they are required:

Label	Input Field
User ID*	Text input field
Last Name*	Text input field
Common Name*	Text input field
Organization Unit*	Text input field
Distinguished Name	Text input field
Object Class	Text input field
Request ID	Text input field

Index

A

access control, AR System 35
accessing, plug-ins 122
active link actions
 DDE 289, 294, 298
 OLE Automation 280
active links, execution order 107
advisory mode 339
aliases, plug-in 137
Apache AXIS, WSDL file 81
API
 AR System External Authentication
 (AREA) 124
 common plug-ins 120
applications
 licensing 363
 overview of licensing 365
AR System
 access control 35
 active links, described 33
 API, understanding 47
 architecture, described 23
 architecture, terminology 22
 C API
 clients 49
 program structure 49
 using for integration 51
 clients
 communications with server 30
 described 25

communications
 between clients and server 30
 with database servers 30
components 32
data, accessing externally 206
database server, described 29
escalations, described 33
external processes 254
filters, described 33
forms, described 33
Java API
 cloning objects 58
 criteria list objects 58
 criteria objects 58
 exception handling 58
 naming convention 68
 object factories 57
 object installation 69
 overview 56
 program model 56
 program structure 59
menus, described 33
ODBC 210
plug-ins, overview 116
security 35
server, communications with clients 30
server, described 28
XML data 250
XML objects 250
XML, using with AR System API 251

AR System Database Connectivity (ARDBC)
 API 128
 API, calling from ARDBC plug-in 132
 plug-in 127
 vendor forms 127

AR System External Authentication (AREA)
 AREA Hub, setting up 174
 authenticating unregistered users 166
 authentication chaining mode, and 167
 authentication chaining mode, effect on 169
 configuring 165
 cross referencing blank passwords 166
 overview 164
 when to use 165

aradmin
 commands and options 230
 examples 234

architecture
 AR System, described 23
 AR System, terminology 22

ARDBC LDAP
 Configuration form 141
 example
 attaching fields to a vendor form 469
 creating a vendor form 466
 defining a filter 472
 generating a DN 472

ARDBC LDAP plug-in. *See* LDAP

AREA
See AR System External Authentication (AREA)

AREA hub, setting up 174

AREA LDAP
 Configuration form 153
See also LDAP

arimportcmd
 commands and options 242, 245
 examples 240, 246
 importing with mapping 241
 importing without mapping 244
 UNIX and 241

armonitor.conf (armonitor.cfg) file
 SNMP 332

arplugin server 121

arsnmp file 325

attributes
 missing XML 399
 object in directory services 150

authentication chaining mode
 effect on authentication processing 169
 specifying 167

authentication processing
 authenticate unregistered users option 166
 authentication chaining mode, specifying 167
 configuring 165
 cross reference blank passwords option 166
 automation servers (OLE) 281

B

basic web services 87

Bind User 156

BMC Remedy Administrator
 command line interface 229
 commands and options (CLI) 230
 configuration details (CLI) 235
 converting form views (CLI) 235
 described 25
 exporting objects (CLI) 234
 extension objects (CLI) 233
 importing objects (CLI) 234
 internet access through proxy server,
 configuring 85

BMC Remedy Import, command line
 interface 240

BMC Remedy Mid Tier, described 28

BMC Remedy SNMP Agent. *See* SNMP

BMC Remedy User
 command line interface 236
 Crystal Reports 214
 described 25
 macros, third-party applications 303
 ODBC driver installed 211
 path and DDE server name 303

C

C API
 clients 49
 overview 46
 program structure 49

- C API (continued)
- understanding 47
 - using for integration 51
- cert7.db certificate database file 143
- cert8.db certificate database file 143
- certificate database 156
- change history
- source control 358
- choice element, web services 389
- ClearCase integration with AR System source control 345
- command (DDE) 297
- command line interface
- aradmin 229, 230, 234
 - arimportcmd 246
 - Remedy Administrator 229
 - Remedy Import 240
 - Remedy User tool client 236, 237
 - runmacro 238, 240
- complex documents
- filter flow 390
 - hierarchical, web services 386
 - publishing (web services) 423, 440
- configuration files
- arsnmpd 325
 - snmpd 331
- configuring
- applications for licensing 367
 - AR System with internet access through proxy server 82
 - BMC Remedy Administrator for internet access through proxy server 85
 - Crystal Reports 214
 - details in Remedy Administrator (CLI) 235
 - LDAP plug-in 141
 - proxy information, access of 83
 - SNMP 324
 - traps for SNMP 329
- consuming
- complex document, web services 440
 - simple flat document, web services 417
 - web services 105
 - web services, flow 106
 - web services, published on same AR System server 109
- creating
- objects (LDAP) 149
 - set fields from web service filter 106
 - vendor forms 146
 - web services 75, 87
- Crystal Reports
- date/time strings 220
 - join forms 220
 - limitations 220
 - login, AR System 215
 - report fields in 217
 - setting up 214
 - sorting in lists 221
 - using 214
- custom web services 78

D

- data
- importing (UNIX) 240
 - organizing, in directory services 145
 - types, web services 408
- data visualization field
- creating 183
 - creating, on Mid Tier 183
 - Java classes 181
 - overview 178
 - registering 187
 - services and BMC Remedy Mid Tier 178
 - services and clients 180
- databases, LDAP certificate 156
- DDE
- action 294
 - active link values, setting 298
 - active links 289
 - DoExecMacro topic 304
 - examples 304, 307, 312
 - executing macros 289
 - fields, setting values 298
 - item name 288
 - keyword 298
 - macros, samples 307
 - Microsoft Excel integration 307–311
 - Microsoft Word integration 312–315
 - overview 288
 - Remedy User tool macros 303

DDE (continued)

- request operation result syntax 298
- RunMacro function 304
- server name for Remedy User 303
- service name 288
- time-out settings 293
- topic name 288
- win.ini configuration 303

deployable applications

- configuring to license 367
- fixed licenses 364
- floating licenses 364
- license options 364
- licenses on server, applying 369
- licenses to users, applying 371
- licensing (overview) 363
- licensing of forms cannot be reversed 367
- read licenses 364

directives, SNMP

- community-based 327
- user-based 328

directory services

- defined 145
- distinguished name attribute 148
- mapping data 146
- object attributes in 150
- objects in 146
- organizing data 145

Distinguished Name

- AREA LDAP configuration 156
- defined 148
- defining filters for 472
- described 150

document

- web services and style 79

documents

- AR System 312
- complex hierarchical, web services 386
- complex, filter flow 390
- simple, web services 384

DoExecMacro DDE topic 304

dynamic data exchange. *See DDE*

E

EIE

- See Enterprise Integration Engine*
- enforced mode 339
- environment
 - source control 338, 341
- escalations
 - execution order 107
- examples
 - aradmin 234
 - arimportcmd 246
- DDE
 - integration with MS Excel 307
 - integration with MS Word 312
 - program and buffer 304
 - requests, assigning values from 298

OLE automation, using 449

runmacro 240

web service

- with complex document, consuming 440
- with complex document, publishing 423
- with simple flat document,
 - consuming 417
 - with simple flat document, creating 412

execution order

- active links and filters 107

exporting

- definitions to source control 350
- objects (CLI) 234

external processes

- running from AR System 254

F

fields

- See also individual database by name*
- database name, identifying 217, 219
- inetorgperson form, summary 475
- modifying in view and vendor forms 204
- values, setting with DDE request results 298

files

- cert7.db 143
- cert8.db 143

filter plug-ins

- API 127
- overview 126

f
filters

- Distinguished Name, defining for 472
- execution order 107
- flow for complex documents 390
- fixed licenses, deployable applications 364
- flat mapping, XML 403
- floating licenses, deployable applications 364
- form views, converting (CLI) 235
- forms
 - ARDBC LDAP Configuration 141
 - AREA LDAP Configuration 153
 - base, web services 77
 - directory services, building 145
 - licensing in deployable application cannot be reversed 367
 - mapping to collection of objects (LDAP) 146
 - vendor, creating 146

G

- genie help (OLE) 281
- get from source (source control) 360
- get operation type (web services) 378
- get operation type, complex documents 390
- guidelines, command line interface 229

H

- HARDWARE keyword 280, 294
- help genie (OLE) 281
- holiday time. *See* business time
- https, accessing WSDL or web services over 83

I

- import in place
 - definitions (source control) 352
 - get from source (source control) 360
 - get mode (source control) 354
- importing
 - data (UNIX) 240
 - definitions from source control 352
 - external XML schema 405
 - objects (CLI) 234
- inetorgperson
 - object class and LDAP 146
 - summary of fields in example 475
 - vendor form, example 466

i
integration

- areas for 18
- asynchronous 19
- at the AR System client 38
- at the AR System server 39
- at the database 39
- benefits 18
- C API, overview 46
- defined 17
- methods 41
- Microsoft Excel with AR System 307
- Microsoft Word with AR System 312
- points, choosing 40
- real-time 19
- source control with AR System 338
- with AR System 19
- integration point
 - AR System client 38
 - AR System server 39
 - C API 46
 - choosing 40
 - database 39
 - multi-platform issues 40
- integration system vendors (ISVs), licensing applications 363
- internet access through proxy server, configuring 82
- Internet Explorer, accessing proxy setting 83
- ISVs. *See* integration system vendors 363
- item name (DDE) 288

J

- Java API
 - exception handling 58
 - naming convention 68
 - object factories 57
 - object installation 69
 - objects, cloning 58
 - objects, criteria 58
 - objects, criteria list 58
 - overview 56
 - program model 56
 - program structure 59

J
JavaScript

- limitations 264
- Run Process action, running from 263

join forms

- Crystal Reports and 220
- web services 395

K

keywords

- DDE 298
- HARDWARE 280, 294

L

LDAP

- ARDBC LDAP configuration form 141
- AREA LDAP configuration form 153
- AREA plug-in
 - authentication processing 162
 - configuring 159
 - configuring group search 160
 - configuring groups 161
 - installation option 152
 - mapping groups 160
- certificate database 156
- certificate database file 143
- defined 140
- distinguished name 156
- inetorgperson object class 146
- mapping data 146
- Mozilla SDK 143
- object creation 149
- overview of plug-in 140
- performance tips 151
- plug-in, configuring 141
- Request ID field limitations 148
- requirements of plug-in 140
- troubleshooting tips 151
- URL standard 146

licenses

- See also* deployable applications
- application, managing 372
- deployable applications overview 363
- options for deployable applications 364
- overview of application 365

L
limitations

- Crystal Reports 220
- Request ID field with LDAP 148
- web services 110
- WSDL, for consumption 113
- XML schemas 110

line items, web services 388

list of web services, viewing 98

logging in, Crystal Reports and 215

M

macros

- DDE programming 289, 303
- MS Office applications, examples 307
- runmacro utility 237
- third-party applications 303

mapping data

- arimportcmd 241
- LDAP 146

mapping, web services 77

message styles in web services 79

methods (OLE)

- adding 282
- deleting 282

methods of integration 41

Microsoft

- .NET, WSDL file 81
- See also* Internet Explorer
- Access, integration with AR System 221
- Excel, integration with AR System 307
- Word, integration with AR System 312

modes (source control)

- advisory 339
- enforced 339
- get mode 354

modifying

- fields in view and vendor forms 204

Mozilla

- certificate database files 143
- LDAP SDK 143
- proxy settings, accessing 83
- SSL, using 143

N

nillable attributes, web services 387

O

object classes

- defined 149
- inetorgperson 146
- objectclass attribute 149

objects

- adding to source control 349
- checking into source control 357
- checking out from source control 356
- creation, supporting 149
- executable, running source control 361
- extension, in Remedy Administrator (CLI) 233
- history in source control 358
- identifying uniquely 148
- properties, simple XML editing 398
- removing from source control 355
- source control and 340
- status history in source control 359
- status list, refreshing in source control 359
- summary, source control 339
- undoing check out from source control 357
- user information in source control 360
- within directory service 146

ODBC

- clients, compatibility with 214
- Crystal Reports and 214–221
- data sources, adding multiple 211
- driver installed with BMC Remedy User and mid tier 211
- Microsoft Access and 221
- Microsoft Excel and 222
- overview 210

OLE

- active links 283
- Automation active link action 280
- methods, adding 282
- methods, creating 452–461
- sample exercise for 449

operations

- types, web services 376
- web services 77

options

- aradmin 230
- arimportcmd 242, 245
- runmacro 238
- source control 347

P

parameter list (OLE) 282

path (DDE) 296

performance tips for LDAP 151

plug-ins

- aliases, examples 137
- API, ARDBC 132
- API, ARDBC-specific 128
- API, filter-specific 127
- AR System Database Connectivity (ARDBC) 127
- AR System External Authentication (AREA) 122
- ARDBC LDAP 141
- ARDBC, using to create a vendor form 133
- AREA 122
- AREA LDAP 152
 - authentication processing 162
 - configuring 159
 - configuring group search 160
 - configuring groups 161
 - mapping groups 160
- arplugin.log 136
- creating 118
- filter 126
- logging 136
- overview 116
- plug-in server 121
- port numbers 138
- plug-ins,
 - aliases, defining 137
- port numbers for plug-ins 138
- proxy
 - configuration information, accessing 83
 - server, configuring AR System with internet access through 82
- proxy server, configuring web services 82

publishing

- complex document, web services 423
- flow for web services 99
- simple flat document, web services 412
- web services 87

PVCS integration with AR System source control 344

R

read licenses, deployable applications 364
refreshing

- status history in source control 359
- status in file list in source control 359

Remote Procedure Call (RPC) message style, web services 79

reporting

- configuring for DDE 299
- logging in from Crystal Reports 215

Request ID field

- identifying objects uniquely 148
- LDAP limitations 148

retrieving data from another application 260

RPC message styles 79

Run Process action

- client and server processes, and 254
- retrieving data from another application 260
- running a process on the web 263
- running JavaScript on a browser 263
- using to start an external application 255

runmacro

- commands and options 237
- DDE function 304
- examples 240

S

schema

- constructs not supported, XML 112
- XML 78

Secure Sockets Layer. *See* SSL

security, AR System 35

servers

- application licenses, applying 369
- automation (OLE) 281
- DDE name 303

proxy, configuring AR System with internet access 82

service name (DDE) 288

set fields from web service filter, creating 106

set operation type

- complex documents 389
- web services 377

simple documents

- flat, consuming (web services) 417
- flat, publishing (web services) 412
- web services 384

Simple Network Management Protocol. *See* SNMP

simple XML editing

- null, empty, or missing values, handling 399
- object properties 398

SNMP

access control 326

AR System processes monitored 321, 322

armonitor configuration file 330

armonitor file 332

arsnmp file 325

community-based directives 327

configuration 324

configuration files 324

directives 326, 327

overview 320

Remedy MIB 322

Remedy SNMP Agent functions 321

snmpd file 331

starting Remedy SNMP Agent 333

stopping Remedy SNMP Agent 333

system information 326

trap configuration 329

traps 323

troubleshooting 334

user-based directives 328

versions supported 320

SOAP

encoding rules for web services 80

headers and authentication 103

web services 79

source control

adding server objects 349

advisory mode 339

source control (continued)

- checking in server objects 357
- checking out server objects 356
- ClearCase integration 345
- copying to .def file 360
- deleting objects from 355
- enforced mode 339
- environment 338, 341
- executable, running 361
- exporting definitions 350
- get mode 354
- history of server objects 358
- history, viewing 358
- import in place 352, 354, 360
- importing definitions 352
- integration with AR System 338
- latest version 359
- object summary 339
- options 347
- PVCS integration 344
- removing server objects 355
- setting up 341
- status history of 359
- status in file list 359
- undoing check out of server objects 357
- user information 360
- using 337

spreadsheets, in the AR System 307

SSL, using certificates 143

T

third-party applications and macros 303

time-out settings, DDE 293

topic name (DDE) 288

troubleshooting LDAP 151

type library information (OLE) 282

U

UNIX

- arimportcmd 241
- data, importing 240

URLs

- LDAP 146
- protected permissions for WSDL file 98
- WSDL (Apache AXIS) 81

URLs (continued)

- WSDL (Microsoft.NET) 81
- WSDL file 98
- XSD file 405

users

- application licenses, applying 371
- information (source control) 360

V

values, handling null, empty, or missing 399

vendor forms 127

- about 193
- attaching fields, example 469
- creating 146
- creating using ARDBC plug-in 133
- creating, example 466
- example 466
- inetorgperson example 466
- modifying fields 204
- troubleshooting 151
- using 192

version, getting latest from source control 359

view forms

- data types supported 199
- modifying fields 204

viewing, list of available web services 98

views, converting a form (CLI) 235

W

warnings

- back up form data before importing in place 352
- licensing of forms in application cannot be reversed 367
- renaming form during import from source control breaks workflow connection 354
- source control, using in mixed AR System environment 341

web service description language (WSDL), described 80

web services

- accessing, over https 83
- advanced XML editing 404
- architecture 76
- authentication 103

web services (continued)
 base form 77
 basic 78, 87
 basic, creating 87
 choice element 389
 clients, writing 81
 complex documents 386
 Configuration Tool for Remedy mid tier 103
 configuring 82
 configuring a plug-in 84
 configuring BMC Remedy Administrator 85
 consuming 105
 consuming flow 106
 consuming on same AR System server 109
 consuming, flow for 106
 create operation 376
 creating 87
 creating filters for 106
 custom 78
 custom, creating 91
 data types 408
 described 76
 environment, setting up 81
 examples 412
 external XML schema, importing 405
 fetch record
 setting maximum 381
 setting starting record 381
 get operation 378, 390
 internet access through a proxy server 82
 Java Runtime Environment 82
 join forms 395
 limitations 110
 line items 388
 list, viewing 98
 mapping 77, 384, 391
 message styles 79
 messaging protocols 79
 nillable attributes 387
 object properties 398
 operations 77, 95, 376
 overview 76
 proxy server 82
 publishing 87, 99
 saving 98

web services (continued)
 set fields from 106
 set operation 376, 377, 389
 simple documents 384
 simple XML editing 397
 SOAP protocol 79
 WSDL 80
 XML editing 397, 404
 XML schema 78
 XPath function 379
Web Services Description Language (WSDL). *See*
 WSDL
workflow actions
 DDE 294
 OLE Automation 280
WSDL
 accessing, over https 83
 file 80, 91, 98
 limitations for consumption 113
 protected permissions in URL 98
 public permissions in URL 98
 URL to WSDL file 98
 web services implementation 79
 WSDL file (Apache AXIS) 81
 WSDL file (Microsoft.NET) 81

X

XML
 advanced editing 404
 AR System API, and 251
 data and AR System 250
 flat mapping 403
 missing attributes 399
 objects and AR System 250
 schema constructs not supported 112
 schema limitations 110
 schema, web services 78
 simple editing 397
 URL with XML schema 405
 web services, editing 397
 WSDL file 80
XPath function
 web services 379, 381



58467