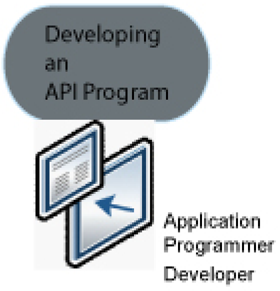
Developing an API program

Author: Lisa Greene

Date: Aug 14, 2013 9:47 AM
URL: https://docs.bmc.com/docs/display/ars81/Developing+an+API+program

Table of Contents

Use the following information to understand how to use C and other compatible languages to write programs that integrate with BMC Remedy AR System. The following information includes a complete reference to the functions of the BMC Remedy AR System C API and also provides an overview of the Java API.



Goal	Instructions
Understand the client-server application model, the BMC Remedy AR System C API program structure, and the library functions.	API overview
Write an API program.	When to use API programming
Learn how to use the BMC Remedy AR System C API for linking with third-party applications, and know the issues and considerations when using the BMC Remedy AR System C API.	BMC Remedy AR System C API overview
Understand the contents of the BMC Remedy AR System C API package, know the C API changes, and the requirements for compiling and linking C API programs.	BMC Remedy AR System C API installation and compilation requirements
Learn about the common elements and functions of the BMC Remedy AR System C API data structures and the relationships among objects.	Data structures
Learn about the supported formats for transforming XML documents to BMC Remedy AR System structures.	Data structures
Understand the various types of BMC Remedy AR System C API functions and their components.	BMC Remedy AR System C API functions
Learn the tips for designing a program, developing a plug-in to send notifications, and enabling a client to perform multiple API calls within a transaction.	Creating and executing BMC Remedy AR System C API programs
Debug your application programs by analyzing the logs and using the driver program.	Debugging API programs
Understand the BMC Remedy AR System plug-in API functions.	BMC Remedy AR System plug-in API functions

Transform BMC Remedy AR System objects into XML format and know the types of functions for object manipulation and transformation.	XML transformation routines
Use dynamic joins, create vendor forms, and retrieve information from forms that store hierarchical data.	Retrieving entries from multiple forms
Learn about the BMC Remedy AR System Java API programming model, understand the run-time configuration to run Java API programs, and develop your BMC Remedy AR System Java client.	BMC Remedy AR System Java API overview

1 API overview

This section explains when to use an application programming interface (API) and provides an overview of the BMC Remedy Action Request System APIs. It includes information that a developer needs to write applications that integrate with BMC Remedy AR System.

This section also includes a complete reference to the functions of the BMC Remedy AR System C API and covers information about the Java API.

Note

The compatibility information listed in the product documentation is subject to change. See Checking system requirements and supported configurations 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.

You must know how to write API programs and be familiar with BMC Remedy AR System, including the overall architecture.

In addition, you should be knowledgeable about the operating system for your environment and have some experience with client/server applications and plug-ins. The information here assumes that you are proficient in C programming and are familiar with arrays, pointers, structure types, and allocated memory. To use the XML import and export functions, you should be knowledgeable about XML.

2 When to use API programming

The primary reason to write your own API program is to satisfy a requirement that you cannot accomplish with the BMC Remedy AR System clients. In addition, API programming gives you flexibility to customize your application.

However, API solutions are more complex to design, implement, and maintain. If you write an API program, you must manage all cross-platform issues and issues related to BMC Remedy AR System upgrades. See Migrating to the current release of BMC Remedy AR System C API for information about changes to the API for version 8.1.

You might want to write an API program if:

- You need to access multiple BMC Remedy AR System components at the same time or integrate with programs or data outside the BMC Remedy AR System.
- You need to perform complex operations that involve multiple forms.
- You need a two-way interface (or gateway) between BMC Remedy AR System and another application.
- The values that you need to specify for **$PROCESS$** or the Run Process action (see Creating and executing BMC Remedy AR System C API programs) exceed the size limitation of the command line. On some systems, for example, the command-line area for **$PROCESS$** or Run Process parameters is limited to 256 bytes (expandable to 4 KB). If you use an API program instead, you can pass a parameter to an API program and manipulate larger-sized data within the program.
- You need to create reports or log files in a format that is not compatible with standard report writing tools.
- You want to create, modify, and delete objects.

3 BMC Remedy AR System C API overview

The BMC Remedy AR System C API provides a common interface client programs, including BMC Remedy AR System clients, can use to communicate with BMC Remedy 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 BMC Remedy AR System C API and how you can use it to integrate BMC Remedy AR System with a third-party product.

Use the information below for details about the BMC Remedy AR System C API.

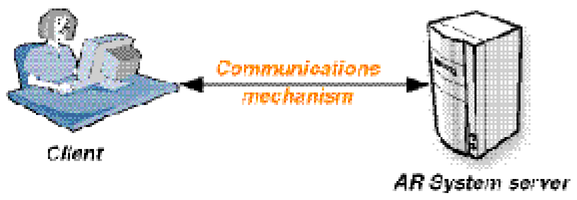
3.1 Client-server application model

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

- A software component with a user interface on a client computer
- 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 computer, which is "serviced" by the server.

Simple client-server model



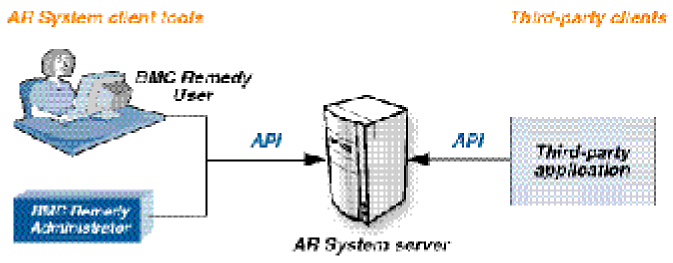
Typically, the client and server components are on different computer systems, and the communications mechanism is based on some form of networking technology, but it is possible to have both the client and server on the same host simply communicating between processes.

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 BMC Remedy AR System server has a fully defined API that is common to all server platforms, both Windows and the UNIX platforms. The BMC Remedy 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 a BMC Remedy AR System server, it can communicate all its requests and receive the replies using this common language. The client does not need to know on which platform the server is running. It also does not need to know about other clients that are using the same server.

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

BMC Remedy AR System API



3.2 BMC Remedy AR System API library functions

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

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

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

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

- **Filters** — Server-side, action-based workflow rules.
- **Escalations** — Server-side, time-based workflow rules.
- **Active links** — Client-side, action-based, and time-based workflow rules.
- **Containers** — Generic lists of references that are used to define guides and applications.
- **Support files** — Files that clients can retrieve. Commonly used for reports but can be any file on the server.

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

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

Additional functions are available for Data Import, Data Export, User Administration, and Connection Control. API functions that deal exclusively with Alert Notification are also available that are mainly used by the legacy BMC Remedy Alert tool, which is no longer shipped with BMC Remedy AR System.

3.3 BMC Remedy AR System C API program structure

The BMC Remedy 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 BMC Remedy 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 `Free`AR functions (or use the free function) to free all allocated memory associated with a specific data structure. For more information, see Freeing allocated memory.
- Shutdown/Cleanup — Call `ARTermination` to perform environment-specific cleanup routines and disconnect from the BMC Remedy AR System servers (required).
  If you use floating licenses and do not disconnect from the server, your license token is unavailable for other users for the defined time-out interval.

3.4 Multithreaded API clients

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

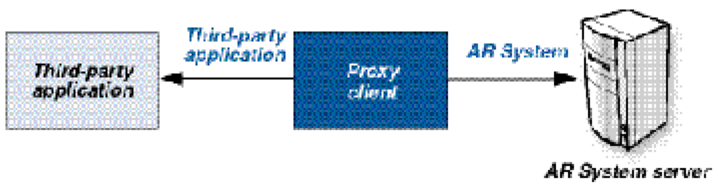
Following is an example of a generic API program source file.

```
{anchor:_Ref375052960}
#include <stdlib.h>
#include <string.h>
#include <ar.h>
#include <arextern.h>
#include <arfree.h>

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

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

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

        /* Remedy Startup */
        if (ARInitialization(&control, &status) >= AR_RETURN_ERROR) {
            printf("\n **** initialization error ****\n");
            exit(-1);
        }
        FreeARStatusList(&status, FALSE);

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

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

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

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

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

    return 0;
}
```

3.5 Using the BMC Remedy AR System API for integration

Typically, the BMC Remedy AR System API is not linked directly into a third-party application. Instead, a separate program is created that interfaces with the BMC Remedy AR System server using the BMC Remedy 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.

Using APIs to link applications



The proxy client that links BMC Remedy AR System with another application does not need to provide all of the features seen in BMC Remedy AR System clients such as BMC Remedy Mid Tier. 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 BMC Remedy AR System server database, with no user interaction. Similarly, a proxy client could monitor the BMC Remedy AR System server 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 BMC Remedy AR System server, the API can be used to integrate with existing BMC Remedy AR System or legacy applications.

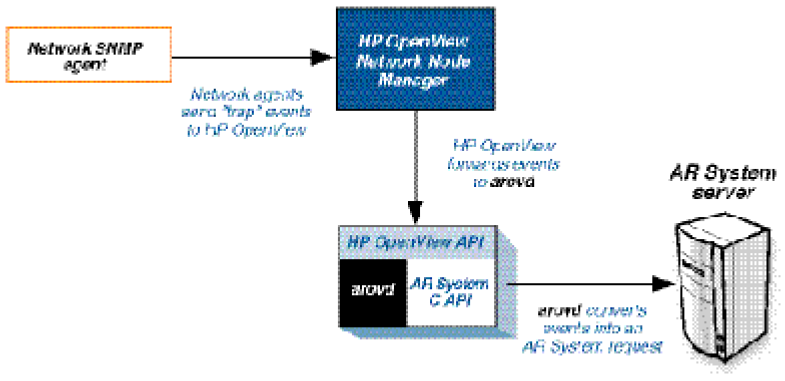
3.5.1 Network management platform integration accessories use case

BMC Remedy makes available a set of accessories that provide integration with network management platforms such as HP OpenView Network Node Manager, IBM NetView for AIX, and Oracle Solstice Domain Manager. The major portion of the integration consists of a set of proxy client applications that take selected events (such as alarms) identified by the management platforms and create trouble ticket records in a BMC Remedy 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 BMC Remedy AR System API (on the other side) to send the information to a BMC Remedy 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 a BMC Remedy AR System server remotely across a network. However, if BMC Remedy 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** (BMC Remedy 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 BMC Remedy AR System API so that events of interest could be translated and stored as records in the BMC Remedy AR System server database.

Integration of BMC Remedy AR System with HP OpenView Network Node Manager


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

3.6 BMC Remedy AR System API issues and considerations

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

- Using the BMC Remedy AR System API requires expertise in C programming. It assumes a technical background and familiarity with the use of compilers.
- The BMC Remedy AR System server is backward compatible, supporting all requests from applications that use the API libraries back to BMC Remedy 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 BMC Remedy AR System API libraries, you will probably need to make changes to your programs. The main program structure and processing, however, need not change.

4 BMC Remedy AR System C API installation and compilation requirements

This section describes the files in the C API package and the setup required to compile and link C API programs. The following topics are provided:

- BMC Remedy AR System C API package contents
- Migrating to the current release of BMC Remedy AR System C API
- Compile and link requirements

4.1 BMC Remedy AR System C API package contents

You can either install the API files when you first install the BMC Remedy AR System server or install them later.

Note

If you have already installed the BMC Remedy AR System server and want to add the API package, make sure that you select the option to *share* the database rather than overwrite it.

The API package includes header files, library files, and source code for the **driver** sample programs. The **driver** utility allows you to manually run individual API functions and see the results returned. For more information, see Using the driver program.

The following topics are provided:

- Header files
- Library files

4.1.1 Header files

The **include** directory contains the following API header files:

File name	Contents
ar.h	Data type and structure definitions, size limits, and constant definitions.
arc.h	Additional structures and constants for C API calls.
ardbc.h	ARDBC definitions.
area.h	AREA definitions.

arerrno.h	Error code definitions.
arextern.h	External declarations for the API functions, specified with and without prototypes for use with standard C, ANSI C, or C++ compilers.
arfilterapi.h	AR Filter definitions.
arfree.h	External declarations for the Free AR functions. These functions recursively free all allocated memory associated with a particular data structure. Like **arextern.h**, declarations are specified with and without prototypes.
arplugin.h	BMC Remedy AR System plug-in definitions.
arstruct.h	Core and reserved field ID definitions, database separator characters, and labels for exporting structure definitions.
artypes.h	BMC Remedy AR System type definition. Included by **ar.h**.

4.1.2 Library files

This section describes the library files that must be available to your API program.

About library files and the 64-bit BMC Remedy AR System server

For the Windows platform, both 32-bit and 64-bit libraries are installed to support 32-bit and 64-bit applications. For the UNIX and Linux platforms, the 32-bit and 64-bit libraries continue to be installed to support 32-bit and 64-bit applications.

HP-UX platform changes

On Itanium platforms, the 32-bit and 64-bit native-mode Itanium libraries are installed. Under HP-UX on Itanium-based HP Integrity servers, all 32-bit API applications now run in native 32-bit mode. Applications compiled using the 64-bit C API run in native 64-bit mode (IA-64) when running under HP-UX on Itanium-based HP Integrity servers.

Custom plug-ins run without change, except for C/C++ plug-ins on the Itanium platform:

- In BMC Remedy AR System 7.1.00, the C plug-in server ran under the HP ARIES emulator and supported 32-bit plug-ins compiled for the PA-RISC platform.
- Beginning with release 7.5.00, BMC Remedy AR System for the Itanium platform does not support PA-RISC plug-ins. If you are upgrading on this platform from release 7.1.00, recompile C/C++ plug-ins as 32-bit Itanium shared libraries.

Linux

When you install the BMC Remedy AR System C and Java APIs on a supported Linux system, most library files are installed in a directory under the server directory, typically, ***arInstallDir*/api/lib**.

The following table lists the installed library files for Linux and which C or Java API programs require them. In the library file names, the placeholder *VerNum* represents the version number of the release. In some cases, this includes a build number. For example, in release 7.6.04, the AR System API file is named **arapi7604jar** or **arapi7604_build001.jar**.

Remember the following considerations for Linux applications:

- The **icu**, **xalan**, and **xerxes** library files are installed in ***arInstalldir***. To make sure your application can access these libraries, copy them to the **lib** directory that contains the other library files.
- To load dynamic libraries on Linux, include the `-ldl link` flag in the `link` command. See the driver sample **makefile**.
- The JNI files and all C API library files are required when a Java API program connects to a pre-7.0.01 BMC Remedy AR System server.

Linux libraries table

Operating system	Required for:	Files
Linux 64-bit	All	libar_lx64.a libar_lx64.so libicudatabmc_lx64.so.32.0 libicui18nbmc_lx64.so.32.0 libicuiobmc_lx64.so.32.0 libiculebmc_lx64.so.32.0 libiculxbmc_lx64.so.32.0 libicutubmc_lx64.so.32.0 libicuucbmc_lx64.so.32.0
	XML functions	libarxmlutil_lx64.so libxalan-cbmc_lx64.so.110.0 libxalanMsgbmc_lx64.so.110.0 libxerces-cbmc_lx64.so.28.0 libxerces-depdombmc_lx64.so.28.0

	Java API	arapiVerNum.jar arapiextVerNum.jar libarjniVerNum.so log4j-1.2.14.jar log4j.xml arsys_sample.xml
Linux 32-bit	All	libar.a libar.so libicudatabmc.so.32.0 libicui18nbmc.so.32.0 libicuiobmc.so.32.0 libiculebmc.so.32.0 libiculxbmc.so.32.0 libicutubmc.so.32.0 libicuucbmc.so.32.0
	XML functions	libarxmlutil.so libxalan-cbmc.so.110.0 libxalanMsgbmc.so.110.0 libxerces-cbmc.so.28.0 libxerces-depdombmc.so.28.0
	Java API	arapiVerNum.jar arapiextVerNum.jar libarjniVerNum.so log4j-1.2.14.jar log4j.xml arsys_sample.xml

UNIX

When you install the BMC Remedy AR System C and Java APIs on a supported UNIX system, most of the library files are installed in a directory under the server directory, typically, ***arInstallDir*/api/lib**.

The following table lists the installed library files for each UNIX platform and what API programs require them. In these file names, the placeholder *VerNum* represents the version number of the release. In some cases, this includes a build number. For example, in release 7.6.02, the AR System API file is named **arapi7602.jar** or **arapi7602_build001.jar**

Remember the following considerations for UNIX applications:

- The **icu**, **xalan**, and **xerxes** library files are installed in ***arInstallDir***. To make sure your application can access these libraries, copy them to the **lib** directory that contains the other library files.
- The name of each 64-bit library file includes a string that indicates its execution mode.
- The JNI files and all C API library files are required when a Java API program connects to a pre-7.0.01 BMC Remedy AR System server.

UNIX library files table

Operating system	**Required for:**	**Files**
HP HP-UX (64-bit Itanium)	All	libar_hpia64.a libar_hpia64.sl libicudatabmc_hpia64.sl.32.0 libicui18nbmc_hpia64.sl.32.0 libicuiobmc_hpia64.sl.32.0 libiculebmc_hpia64.sl.32.0 libiculxbmc_hpia64.sl.32.0 libicutubmc_hpia64.sl.32.0 libicuucbmc_hpia64.sl.32.0
	XML functions	libarxmlutil_hpia64.sl libxalan-cbmc_hpia64.sl.110.0 libxalanMsgbmc_hpia64.sl.110.0 libxerces-cbmc_hpia64.sl.28.0 libxerces-depdombmc_hpia64.sl.28.0

	Java API	libarjniVerNum.sl arapiVerNum.jar arapiextVerNum.jar log4j-1.2.14.jar log4j.xml arsys_sample.xml
HP HP-UX (32-bit Itanium)	All	libar_hpia32.sl libar_hpia32.a libicudatabmc_hpia32.sl.32.0 libicui18nbmc_hpia32.sl.32.0 libicuiobmc_hpia32.sl.32.0 libiculebmc_hpia32.sl.32.0 libiculxbmc_hpia32.sl.32.0 libicutubmc_hpia32.sl.32.0 libicuucbmc_hpia32.sl.32.0
	XML functions	libarxmlutil_hpia32.sl libxalan-cbmc_hpia32.sl.110.0 libxalanMsgbmc_hpia32.sl.110.0 libxerces-cbmc_hpia32.sl.28.0 libxerces-depdombmc_hpia32.sl.28.0
	Java API	libarjniVerNum_hpia32.so arapiVerNum.jar arapiextVerNum.jar log4j-1.2.14.jar log4j.xml arsys_sample.xml

IBM AIX (64-bit)	All	libar_aixp64.a libicudatabmc_aixp6432.0.a libicui18nbmc_aixp6432.0.a libicuiobmc_aixp6432.0.a libiculebmc_aixp6432.0.a libiculxbmc_aixp6432.0.a libicutubmc_aixp6432.0.a libicuucbmc32_aixp64.a libicuucbmc_aixp6432.0.a
	XML functions	libarxmlutil_aixp64.a libxalan-cbmc_aixp64110.0.a libxalanMsgbmc_aixp64110.0.a libxerces-cbmc_aixp6428.0.a libxerces-depdombmc_aixp6428.0.a
	Java API	libarjniVerNum.a arapiVerNum.jar arapiextVerNum.jar log4j-1.2.14.jar log4j.xml arsys_sample.xml
IBM AIX (32-bit)	All	libar.a libicudatabmc32.0.a libicui18nbmc32.0.a libicuiobmc32.0.a libiculebmc32.0.a libiculxbmc32.0.a libicutubmc32.0.a libicuucbmc32.0.a

	XML functions	libarxmlutil.a libxalan-cbmc110.0.a libxalanMsg110.0.a libxerces-cbmc28.0.a libxerces-depdombmc28.0.a
	Java API	libarjniVerNum.a arapiVerNum.jar arapiextVerNum.jar log4j-1.2.14.jar log4j.xml arsys_sample.xml
Oracle Solaris* (64-bit)	All	libar_solsp64.a libar_solsp64.so libicudatabmc_solsp64.so.32.0 libicui18nbmc_solsp64.so.32.0 libicuiobmc_solsp64.so.32.0 libiculebmc_solsp64.so.32.0 libiculxbmc_solsp64.so.32.0 libicutubmc_solsp64.so.32.0 libicuucbmc_solsp64.so.32.0
	XML functions	libarxmlutil_solsp64.so libxalan-cbmc_solsp64.so.110.0 libxalanMsgbmc_solsp64.so.110.0 libxerces-cbmc_solsp64.so.28.0 libxerces-depdombmc_solsp64.so.28.0

	Java API	libarjniVerNum.so arapiVerNum.jar arapiextVerNum.jar log4j-1.2.14.jar log4j.xml arsys_sample.xml
Oracle Solaris* (32-bit)	All	libar.a libar.so libicudatabmc.so.32.0 libicui18nbmc.so.32.0 libicuiobmc.so.32.0 libiculebmc.so.32.0 libiculxbmc.so.32.0 libicutubmc.so.32.0 libicuucbmc.so.32.0
	XML functions	libarxmlutil.so libxalan-cbmc.so.110.0 libxalanMsgbmc.so.110.0 libxerces-cbmc.so.28.0 libxerces-depdombmc.so.28.0
	Java API	libarjniVerNum.so arapiVerNum.jar arapiextVerNum.jar log4j-1.2.14.jar log4j.xmarsys_sample.xml
* To load dynamic Solaris libraries, include the `-ldl link` flag in the `link` command. See the driver sample **makefile**.		

At program run-time, all required library files must be in an accessible directory. Add the full path of the directory or directories to the environment variable listed in the following table.

Library path environment variables table

Operating system	Environment variable
IBM AIX	LIBPATH
HP HP-UX (Itanium)	SHLIB_PATH
Linux	LD_LIBRARY_PATH
Oracle Solaris	LD_LIBRARY_PATH

Windows

When you install the BMC Remedy AR System C and Java API on Windows, most library files are installed in a directory under the server directory, typically ***arInstalldir**serverName*\ARServer\Api\lib**.

The following table lists the installed library files and what API programs require them. In these file names, the placeholder *VerNum* represents the version number of the release. In some cases, this includes a build number. For example, in release 7.6.02, the AR System API **.dll** file is named **arapi7602.dll** or **arapi7602_build001.dll.**

Remember the following considerations for Windows applications:

- The **icu** library files are installed in ***arInstalldir***. To make sure your application can access these libraries, copy them to the **lib** directory that contains the other library files.
- The JNI files and all C API library files are required when a Java API program connects to a pre-7.0.01 BMC Remedy AR System server.

Windows library files table

Operating system	Required for:	Files
Windows (32-bit)	All	icudt32.dll icuinbmc32.dll icuucbmc32.dll
	C	arapiVerNum.dll arapiVerNum.lib arrpcVerNum.dll arutlVerNum.dll jlicapiVerNum.dll
	C XML functions	arxmlutilVerNum.dll

	Java API	arjniVerNum.dll arapiVerNum.jar arapiextVerNum.jar jlicapiVerNum.jar log4j-1.2.14.jar log4j.xml arsys_sample.xml
Windows (64-bit)	All	icudt32.dll icuinbmc32_win64.dll icuucbmc32_win64.dll
	C	arapiVerNum_win64.dll arapiVerNum_win64.lib arrpcVerNum_win64.dll arutlVerNum_win64.dll
	C XML functions	arxmlutilVerNum_win64.dll
	Java API	arjniVerNum.dll arapiVerNum.jar arapiextVerNum.jar jlicapiVerNum.jar log4j-1.2.14.jar log4j.xml arsys_sample.xml

In addition to these files, your program needs access to **arcatalog_eng.dll** or the message catalog for your locale. The message catalogs are installed in the server directory, typically, **C:\Program Files\BMC Software\ARSystem**.

At program run-time, all required library files must be in an accessible directory. Add the full path of the directory or directories to the Windows **PATH** environment variable.

4.2 Migrating to the current release of BMC Remedy AR System C API

This section summarizes the C API changes from the 7.5 release through the 8.1 release of BMC Remedy AR System.

BMC Remedy AR System 8.1 is backward-compatible, supporting all requests from client applications that use the 7.6, 7.5, 7.1, 7.0, 6.3, 6.0, 5.1, 5.0, 4.x, or 3.2 API libraries. 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 BMC Remedy AR System 8.1 servers.

If you link to the 8.1 API libraries, you must make changes to your existing programs and recompile them. The main program structure and processing, however, need not change.

Some API changes that were made as of release 8.1 might affect your existing API programs compiled with API releases previous to 8.1. Use the following information to understand the changes and additions to the API.

- Changed and new parameters
- Data structure changes and additions

4.2.1 Changed and new parameters

The following table lists the C API functions with new or changed parameters as of the 8.1 release:

Affected functions	**Parameter**	**Change or addition**
`ARGetListEntry` `WithMultiSchemaFields`	`getListFields`	Now is an `ARMultiSchemaFieldFuncList`
	`queryFromList`	Now is an `ARMultiSchemaFuncQueryFromL`
	`groupby`	New parameter which is an `ARMultiSchemaFieldIdList`
	`having`	New parameter which is an `ARMultiSchemaFuncQualifierS`
	`entryList`	Now is an `ARMultiSchemaFieldFuncValue`
`ARGetContainer` `ARGetSchema` `ARGetField` `ARGetActiveLink`	`assignedGroupList`	New parameter to support hierarchic

`ARGetMultipleContainers`
`ARGetMultipleSchemas`
`ARGetMultipleFields`
`ARGetMultipleActiveLinks`
	`assignedGroupListList`	New parameter to support hierarchic

`ARExport`	`exportOption`	New parameter to support object exp

`ARCreateSchema`
`ARSetSchema`
`ARDeleteSchema`
`ARCreateCharMenu`
`ARSetCharMenu`
`ARDeleteCharMenu`
`ARCreateActiveLink`
`ARSetActiveLink`
`ARDeleteActiveLink`
`ARCreateEscalation`
`ARSetEscalation`
`ARDeleteEscalation`
`ARCreateFilter`
`ARSetFilter`
`ARDeleteFilter`
`ARCreateContainer`
`ARSetContainer`
`ARDeleteContainer`
`ARImport`
`ARSetImage`
`ARCreateImage`
`ARDeleteImage`
	`objectModificationLogLabel`	New parameter added to support ve control.

`ARMergeEntry`	`query`, `multimatchOption`	Two new parameters added.

`ARGetField`
`ARSetField`
`ARCreateField`
`ARGetListField`
`ARGetVUI`
`ARSetVUI`
`ARCreateVUI`
`ARGetListVUI`
`ARGetListImage`
	`objPropList`	New parameter added.

`ARGetMultipleFields` `ARSetMultipleFields`	`objPropListList`	New parameter added.

`ARCreateMultipleFields` `ARGetMultipleVUIs`		
`ARExpandCharMenu`	`maxRetrieve`, `numMatches`	Two new parameters added.
`ARGetDSOMappingFromXML` `ARSetDSOMappingToXML`	`excludeFlds`, `rtnExcludeFlds`	Two new parameters added.

4.2.2 Data structure changes and additions

This section describes data structure additions and changes.

New structures

The following new structures support the **groupby**, **having**, and **aggregate** functionality added to `ARGetListEntryWithMultiSchemaFields`:

- `ARMultiSchemaFuncQueryFromList` (replaces `ARMultiSchemaQueryFromList`)
- `ARMultiSchemaFuncQueryFromStruct` (replaces `ARMultiSchemaQueryFromStruct`)
- `ARMultiSchemaRecursiveFuncQueryStruct` (replaces `ARMultiSchemaRecursiveQueryStruct`)
- `ARMultiSchemaNestedFuncQueryStruct` (replaces `ARMultiSchemaNestedQueryStruct`)
- `ARMultiSchemaFieldFuncList`
- `ARMultiSchemaFieldFuncStruct`
- `ARMultiSchemaFuncQualifierStruct`
- `ARMultiSchemaFuncAndOrStruct`
- `ARMultiSchemaFuncRelOpStruct`
- `ARMultiSchemaFieldFuncValueOrArithStruct` (replaces `ARMultiSchemaFieldValueOrArithStruct`)
- `ARMultiSchemaFuncArithOpStruct`
- `ARMultiSchemaFuncValueSetQueryStruct`
- `ARMultiSchemaFuncCurrencyPartStruct` (replaces `ARMultiSchemaCurrencyPartStruct`)
- `ARMultiSchemaFuncStatHistoryValue` (replaces `ARMultiSchemaStatHistoryValue`)
- `ARMultiSchemaFieldFuncValueStruct` (replaces `ARMultiSchemaFieldValueStruct`)
- `ARMultiSchemaFieldFuncValueList` (replaces `ARMultiSchemaFieldValueList`)
- `ARMultiSchemaFieldFuncValueListList` (replaces `ARMultiSchemaFieldValueListList`)

If your application uses `ARGetListEntryWithMultiSchemaFields`, you will need recompile it after linking to the release 7.6.04 API library. For more information about the changes to this function, see ARGetListEntryWithMultiSchemaFields, Structures for ARGetListEntryWithMultiSchemaFields, and Using aggregate functions, groupBy clauses, and having clauses.

If an `ARGetListEntryWithMultiSchemaFields` API call that uses the new options for this function is passed to a pre-7.6.02 BMC Remedy AR System server, the new options are stripped from the query.

4.3 Compile and link requirements

This section covers the setup required to compile and link C API programs. The following topics are provided:

- Compiler information
- Link information
- Working with makefiles

4.3.1 Compiler information

For compilers that require strict ANSI, define **AR_STRICT_ANSI** in your code and use **notQual** in place of **not** to reference the member of the structures of type `ARQualifierStruct`. This resolves compile errors such as the following:

```
Line xxx from ar.h   "Alternative token 'not' may not be
used as an identifier" struct ARQualifierStruct *not;
```

This code from **ar.h** shows how **AR_STRICT_ANSI** is used:

```
typedef struct ARQualifierStruct
   {
      unsigned int                     operation;
      union
      {
         ARAndOrStruct                 andor;
         #ifndef AR_STRICT_ANSI
            struct ARQualifierStruct  *not;
         #else
            struct ARQualifierStruct  *notQual;
         #endif
         ARRelOpStruct                *relOp;
         ARInternalId                  fieldId;
      } u;
   }  ARQualifierStruct;
```

Linux and UNIX

Compile your program using an ANSI standard C or C++ compiler.

Windows

Compile your program with Microsoft Visual C++ .NET 2003 (version 7.1).

Configure the compiler as follows:

- Structure member alignment: **8 bytes** (default)

- Code generation: **Multithreaded DLL**. Do not use **Debug Multithreaded DLL** (**/MD**). Where other included libraries cause conflicts, add **/nodefaultlib:"MSVCRTD"** to the project options to avoid using the Debug C run-time library. If your program references this library at run time, memory management errors occur when memory pointers are referenced by both the Debug and Release C run-time libraries.

4.3.2 Link information

The following table lists the required API libraries. See Linux library files table, UNIX library files table, or Windows library files table for specific library file names.

.API libraries required for linking

Platform	Library
Windows	**arapi*VerNum*.lib** **arapi*VerNum*.dll** (required at runtime) **arutl*VerNum*.dll** (required at runtime)
Solaris, Linux	**libar.a** (archive) **libar.so** (shared library)
HP-UX	**libar.a** (archive) **libar.sl** (shared library)
AIX	**libar.a** (archive) **libar.so** (shared library)

In addition, if the **icu** library files do not appear in the ***arInstallDir*/api/lib** directory, copy them to that location from **arInstalldir**.

In general, library files names contain a suffix that indicates the platform. (This does not apply to some 32-bit library file names.) The following table provides example file names for several platforms:

Platform	Example file name
HPUX Itanium 32-bit	**libar_hpia32.sl**
HPUX Itanium 64-bit	**libar_hpia64.sl**
HPUX PA-RISC 64-bit	**libar_hppa64.sl**
Linux 64-bit	**libar_lx64.so**
AIX 64-bit	**libar_aixp64.a**
Solaris 64-bit	**libar_solsp64.so**

The macro **$(LARCH)** is set to the needed suffix in each per-platform makefile.

4.3.3 Working with makefiles

While makefile content is generally the same in both Windows and UNIX environments, the process of creating and editing makefiles is different. The **driver** program source code includes makefiles that you can use as templates for creating your own makefiles.

Linux and UNIX

The following sample makefile for the **driver** program shows a typical set of flags in the Linux and UNIX environments.

```
#
# This Makefile builds 'driver' as a 32-bit SPARC binary
# for Solaris.
#

#
# Parameters.

PROGRAM   = driver
SOURCES   = api.c get.c main.c print.c util.c
OBJECTS   = api.o get.o main.o print.o util.o
LARCH     =

# Compiler flags.

CC = cc
CFLAGS   = -g -DDEBUG -D_REENTRANT
CPPFLAGS = -I../include -I../../include
LDFLAGS  = -L../lib -L../../lib
LDLIBS   = -lar$(LARCH) -lnsl -lw -lpthread -lcurses -ldl

# Standard targets.

all:     $(PROGRAM)

objects: $(OBJECTS)

$(PROGRAM): $(OBJECTS)
      $(CC) -o $(PROGRAM) $(OBJECTS) $(LDFLAGS) $(LDLIBS)

clean:
      $(RM) $(PROGRAM) $(OBJECTS) core
```

Note

See your system manual to find out if you must place library files in a specific order in the makefile. With Solaris, you must include the **lpthread** switch for the makefile line that starts with **LDLIBS**.

Windows

Creating and editing makefiles in the Windows environment is considerably simpler. The Microsoft Visual C++ compiler creates makefiles for you and provides a graphical interface for specifying the options that you want. You specify the appropriate makefile when you compile your program:

```
nmake /f filename.mak
```

The **sample.mak** file that comes with the **driver** program contains a sample makefile for Windows.

5 Data structures

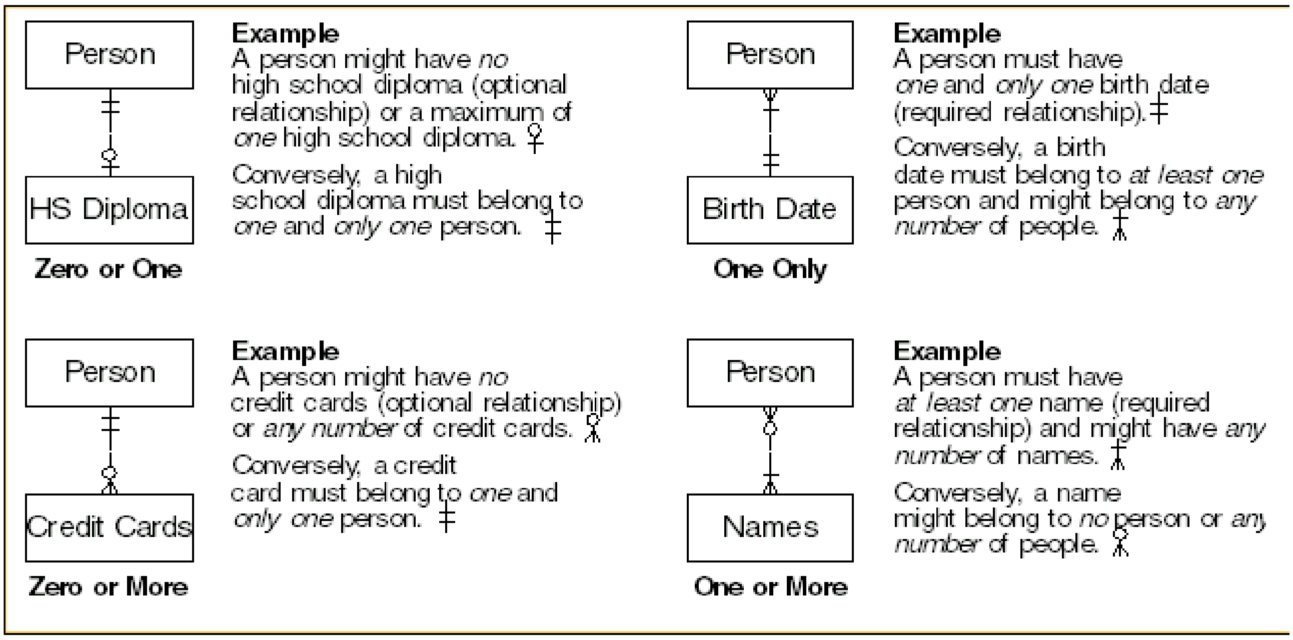
This section describes the most common elements and functions of BMC Remedy AR System C API key data structures. See the header files listed in Using the driver program for complete data type and structure specifications. The following topics are provided:

5.1 Data relationships

The BMC Remedy AR System C API contains structures that store both simple and complex information. Structures that store simple information, such as the type of value or the product of an arithmetic operation, serve as building blocks for complex structures.

This section contains data structure diagrams that explain these hierarchical relationships. The diagrams also identify which structures or groups of structures BMC Remedy AR System uses to manipulate objects.

The following figure explains the data relationship notation.



Example
A person might have *no* high school diploma (optional relationship) or a maximum of *one* high school diploma.
Conversely, a high school diploma must belong to *one* and *only one* person.

Example
A person must have *one* and *only one* birth date (required relationship).
Conversely, a birth date must belong to *at least one* person and might belong to *any number* of people.

Example
A person might have *no* credit cards (optional relationship) or *any number* of credit cards.
Conversely, a credit card must belong to *one* and *only one* person.

Example
A person must have *at least one* name (required relationship) and might have *any number* of names.
Conversely, a name might belong to *no* person or *any number* of people.

Note

For all data structure diagrams in this section, lack of notation indicates a one only relationship.

5.2 C data types

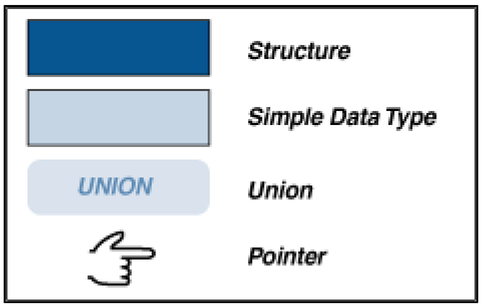
The BMC Remedy AR System API uses simple C data types and these BMC Remedy AR System data types:

Data type	Description
typedef int ARLong32 or typedef long ARLong32	32-bit signed integer for either 32-bit or 64-bit computers. This data type is exactly 32 bits and replaces the long data type in the C API. See the **artypes.h** file.
typedef unsigned int ARULong32 or typedef unsigned long ARLong32	32-bit unsigned integer for either 32-bit or 64-bit computers. This data type is exactly 32 bits and replaces the unsigned long data type in the C API. See the **artypes.h** file.
typedef unsigned char ARBoolean	Boolean flag set to TRUE or FALSE.
typedef char ARAccessNameType[AR_MAX_ACCESS_NAME_SIZE +1]	Structure that holds a user or group name or password.
typedef char ARAuthType[AR_MAX_AUTH_SIZE + 1]	Structure that holds an authentication string.
	Structure that holds an entry identification value.

typedef char AREntryIdType[AR_MAX_ENTRYID_SIZE + 1]	
typedef ARLong32 ARInternalId	Structure that holds an internal identification value.
typedef char ARLocaleType[AR_MAX_LANG_SIZE + 1]	Locale; language-country.
typedef char ARNameType[AR_MAX_NAME_SIZE + 1]	Structure that holds object names.
typedef char ARPasswordType[AR_MAX_PASSWORD_SIZE + 1]	Structure that holds a password.
typedef char ARServerNameType[AR_MAX_SERVER_SIZE + 1]	Structure that holds a server name.
typedef ARLong32 ARTimestamp	UNIX-style time stamp; seconds since January 1, 1970.

The legend in the following figure applies to all data structure diagrams in this section. For simplicity, the data types defined here are considered simple types.

Legend for data structure diagrams



Unless otherwise specified, see the **ar.h** file for complete structure and variable definitions.

5.3 Lists and structures

Lists manipulate sets of objects of a particular type, for example:

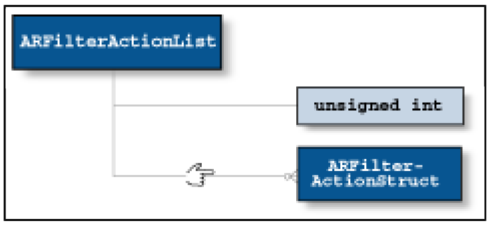
- `ARNameList` manipulates any set of names.
- `ARDiaryList` manages the group of entries in a diary field.
- `AREntryListList` represents any group of entries in a schema.
- `ARFilterActionList` defines the set of actions that the server performs when it executes a filter or escalation.

While each type of list has its own data structure, all list structures have the same form:

```
typedef struct {
    unsigned int     numItems;
    AR xxxStruct      *xxxList;
} ARxxxList;
```

Each list structure has an integer that identifies the number of items in the list and a pointer to the allocated memory containing those items. If the number of items is zero, the server or client ignores the pointer and the pointer is usually set to `NULL`. The following figure shows an example that uses the `ARFilterActionList` structure.

Generic list structure



Note

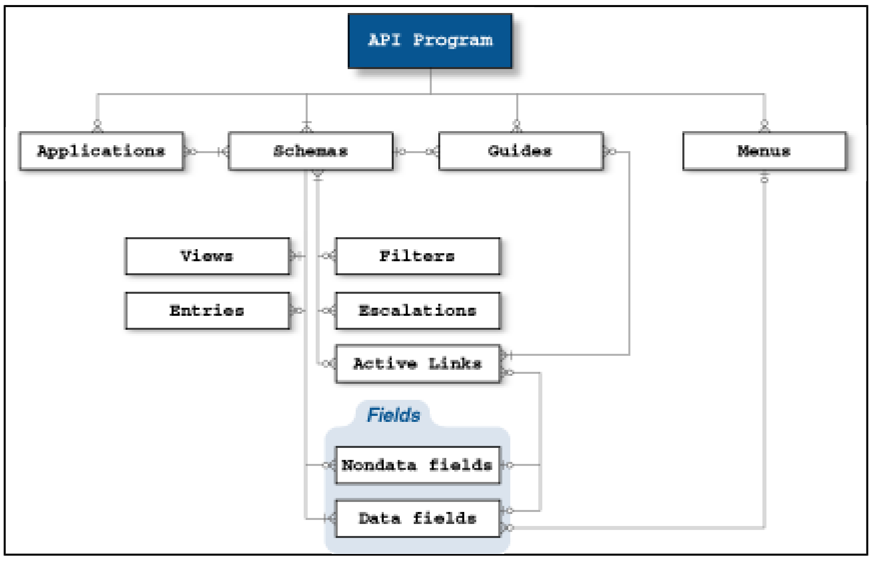
For simplicity, subsequent structure diagrams do not illustrate the **numItems** element (unsigned int) for any list structures. The number of items in the list is the first component of all list structures in the API.

In general, the BMC Remedy AR System server handles lists as arrays instead of as linked lists. It then stores all items in the structure in a single area of memory with a pointer that identifies the memory address of the first item in the array. If the item structure itself (ARFilterActionStruct) contains a pointer, the server allocates the nested memory separately.

5.4 High-level object relationships

To understand specific data structures, you should understand the high-level relationships between BMC Remedy AR System objects. (For descriptions of these objects, see Workflow objects.) As the following figure shows, the foundation of all client programs built on BMC Remedy AR System is the schema. Although the BMC Remedy AR System API uses schemas, the BMC Remedy AR System clients see schemas as forms. A client must access at least one schema and can access many.

High-Level object relationships



Note

For information about the notation used to represent data relationships, see Data relationships.

Your client program can also have menus and **containers**. Either servers or schemas (forms) can own containers, which the server uses to create guides and **applications**. Applications can contain one to many schemas. Guides can be shared and can contain one to many active links.

AR System has the following types of schemas:

- **Data** — The most common. Data schemas contain data in the BMC Remedy AR System database.
- **Display-only** — Show data from one data schema. Display-only schemas do not contain data.
- **Join** — Show data from two or more data schemas with a common matching field. Join schemas do not contain data. When you modify values in a join schema, you are changing the data in its parent data schemas.
- **Vendor** — Acquire data from a plug-in service and act like data schemas.
- **View** — Show data from external database tables and act like data schemas.

Each schema must have at least one **view** (the default view) but can have many.

Each data schema must have at least one **field** but is likely to have many fields. (By default, every data schema is created with nine core fields that cannot be deleted. For more information about core fields, see Core fields.)

AR System has the following types of fields:

- **Data** — A schema must have at least one data field.
- **Nondata** — Trim, control, panel, and table. Nondata fields are optional.

Most data schemas have many **entries**, but a schema can exist with no entry. Although the BMC Remedy AR System API still uses this term, the BMC Remedy AR System clients use the term **requests**.

Any number of **filters**, **escalations**, or **active links** can be associated with a schema or list of schemas, but these objects are not required. See Schemas and structures.

Active links can be associated with either data or control fields. An active link must be associated with a particular schema. A data or control field can have any number of active links associated with it. A particular execute-on condition of an active link can be associated with only one field. Different execute-on conditions in the same active link can reference different fields.

Menus can be associated only with data fields. A menu can be associated with *any number* of data fields, both in a schema and across multiple schemas. However, a data field can have only *one* menu associated with it.

5.5 Login and session information

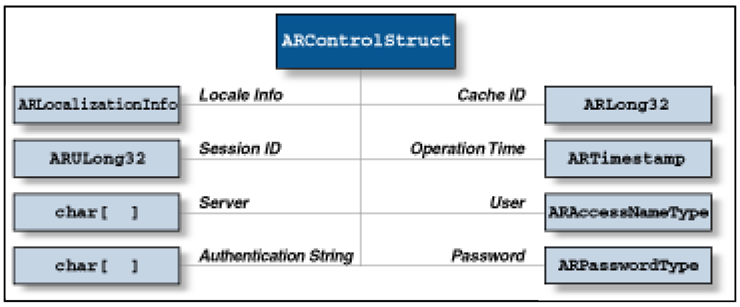
Almost every BMC Remedy AR System C API function has a `control` parameter as its first input argument. This parameter contains the login and session information required to connect to an BMC Remedy AR System server and, thus, required for almost every operation.

Note

The control parameter was optional in BMC Remedy AR System version 3.x and earlier. Therefore, you must add the control parameter to recompiled pre-4.x API programs if you use these programs with later versions of the BMC Remedy AR System API.

The **control** parameter is a pointer to an `ARControlStruct` structure (see the following figure).

Structure used to provide required login information



This structure has the following elements:

`Cache ID`	Identifier of the cache area allocated by the server to store user information. Before the first API call, initialize it to zero. The server assigns this value; do not change it. It enables the server to use the cache instead of reloading user information for each API call.

`Operation Time`	Timestamp specifying the date and time the operation occurred on the server. The server assigns this value for each API call.
`User`	(Required) Login name to connect to the server. The privileges associated with the user determine whether the API function call can be performed.
`Password`	(Required) Password for the specified user name, in clear text. The API encrypts this parameter before sending it to the server.
`Locale Information`	(Required) String that specifies the language for returning error messages (if a message catalog exists), formatting date and time information, sorting or comparing values, and locale information. The locale information is in the following form: `language[_territory[.codeset]][@modifier]` For example: `en_US.ISO8859-15@euro` The string must match the language strings used by the `setlocale` function. The default is a C or an empty string.
`Session ID`	Identifier for the session control block, a structure containing all data needed to maintain the state of API operations. Each API session has a unique session control block, created when you initialize the session by calling `ARInitialization`.
`Server`	(Required) String specifying the name of the server to connect to.
`Authentication String`	Client-provided authentication string, such as the domain.

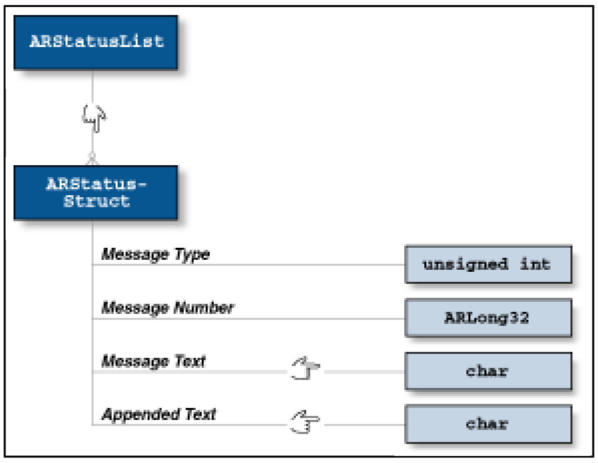
Nearly all function calls require the login and session information that `ARControlStruct` contains (stored in both single- and multiple-server environments) because the API does not always maintain a server connection between calls.

When a program calls `ARInitialization` at the beginning of its execution, the BMC Remedy AR System C API returns the data in a `ARControlStruct`. This is the structure that the program passes as an input parameter in subsequent API calls.

5.6 Status information

The last return value of almost all API functions is a `status` parameter. This parameter contains information about the success or failure of the call. The `status` variable is a pointer to a structure (see the following figure).

Structure used to return function status



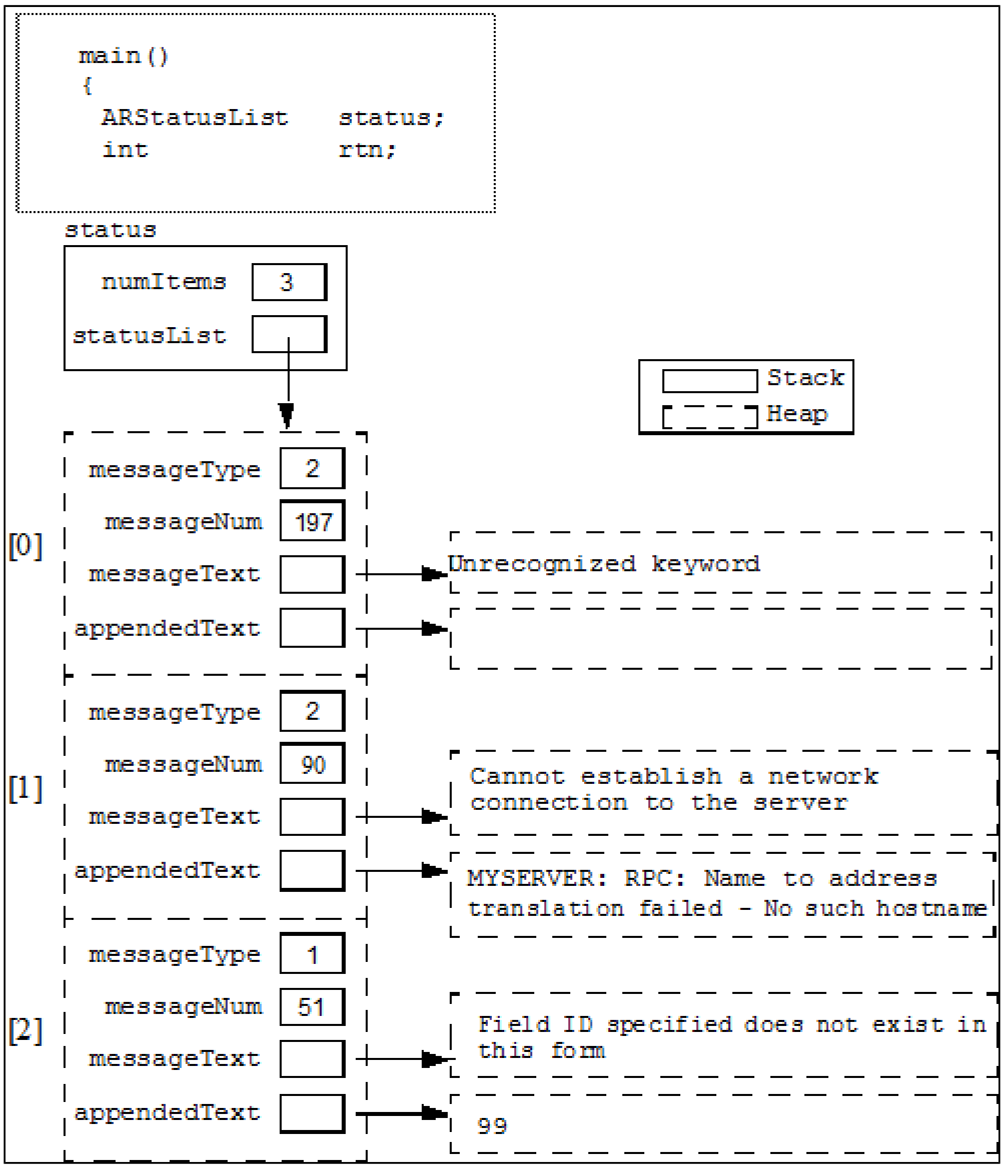
Like all list structures in BMC Remedy AR System (see Lists and structures), `ARStatusList` has zero or more `ARStatusStruct` items. Each `ARStatusStruct` item represents a message that the function call generates and has the following elements:

`Message Type`	Integer that specifies the type of message being returned (see the following table).
`Message Number`	Error number associated with the message (defined in **arerrno.h**).
`Message Text`	Message text in the language that the `ARControlStruct` specifies (if possible). To get the text of the message in the local language, call `ARGetTextForErrorMessage`.
`Appended Text`	Text that augments the message text. This text might come from the BMC Remedy AR System server, operating system, or database management system. The `AR_MAX_MESSAGE_SIZE` limits the length to 255 bytes.

0	AR_RETURN_OK	Operation successful — status might contain one or more *informational* messages.
1	AR_RETURN_WARNING	Operation successful, but some problems encountered — status contains one or more *warning* messages and might also contain informational messages.
2	AR_RETURN_ERROR	Operation failed — status contains one or more *error* messages and might also contain warning or informational messages.
3	AR_RETURN_FATAL	Operation failed — status might contain one or more messages.
4	AR_RETURN_BAD_STATUS	Invalid status parameter — operation cancelled.
5	AR_RETURN_PROMPT	Status for the active link action.
6	AR_RETURN_ACCESSIBLE	Status message for client accessibility.

The following figure shows an example of the layout of the `ARStatusList` structure.

Example of `ARStatusList` structure



In this example, the `status` parameter is defined as a stack variable of type `ARStatusList` and has an integer that specifies the number of messages in the list (three in this case) and a pointer to their location on the heap. Based on the size and number of the returned messages, the system dynamically allocates the memory space.

The messages themselves are stored in an array of `ARStatusStruct` structures, each containing the message type, message number, and pointers to the message text and any appended text. The messages are sorted in decreasing order of severity, based on the message type value (see Error checking). The return value for the function is the message type associated with the first (most recent and most serious) message in the list. In this example, the return value is 2.

To access individual elements in the list, use standard array notation. For example, `status.statusList.messageText` identifies the message text associated with the first message, and `status.statusList.messageNum` provides the message number for the second message

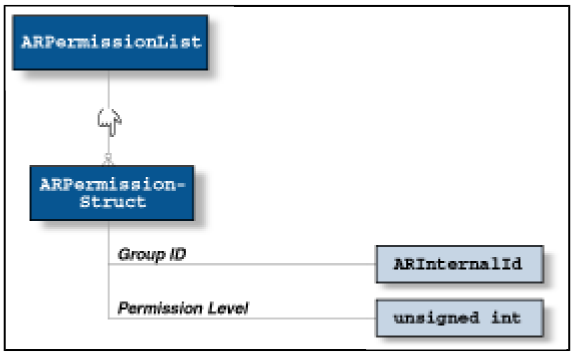
After a program processes a returned status list, it must call `FreeARStatusList` to free any allocated memory. If `numItems` equals zero, the call is optional.

5.7 Permissions and structures

Permission for an BMC Remedy AR System object is generally defined by specifying the list of groups that can access it (use the `ARInternalIdList` structure). The `groupList` parameter for the active link functions is of this type.

For schemas, fields, and containers, however, you can assign different levels of access to different groups. Permission for these objects is defined by the `groupList` (schemas and containers) and `permissions` (fields) parameters, both of which are pointers to structures of type `ARPermissionList` (see the following figure).

Structures used to define field and schema permissions



Each `ARPermission` in the list represents a particular access control group and has the following elements:

Group ID	Internal ID of the group.
Permission Level	Integer that specifies the access privileges for the group. The following tables list the valid integers for schemas, containers, and fields.

Schema/Container permission values for `ARPermissionStruct`

0	AR_PERMISSIONS_NONE	Group has no access to the schema or container.
1	AR_PERMISSIONS_VISIBLE	Group can see the schema or container.
2	AR_PERMISSIONS_HIDDEN	Group cannot see the schema or container.

Field permission values for `ARPermissionStruct`

0	AR_PERMISSIONS_NONE	Group has no access to the field.
1	AR_PERMISSIONS_VIEW	Group has read-only access to the field.
2	AR_PERMISSIONS_CHANGE	Group has read-write access to the field.

Depending on whether you are assigning permission to a schema or a field, different privileges are associated with the possible values.

5.8 Group information and structures

The structures `ARGroupInfoList` and `ARGroupInfoStruct` are used the `ARGetListGroups` API function.

5.8.1 ARGroupInfoList

A list of zero or more groups.

```
typedef struct ARGroupInfoList
{
   unsigned int        numItems;
   ARGroupInfoStruct  *groupList;
}  ARGroupInfoList;
```

This structure has these elements:

Element	Description
`numItems`	Integer that specifies the number of items in the list
`groupList`	Pointer to each item in the list. Items are represented by `ARGroupInfoStruct`

5.8.2 ARGroupInfoStruct

```
typedef struct ARGroupInfoStruct
{
   ARInternalId       groupId;
   unsigned int       groupType;
   ARAccessNameList   groupName;
   unsigned int       groupCategory;
   ARInternalId       groupParent;     /* parent of this group
                                          (may be none) */
} ARGroupInfoStruct;
```

This structure has these elements:

Element	Description
`groupId`	The group number
`groupType`	The group type (none, view, or change)
`groupName`	The name of the group.
`groupCategory`	Category of the group (regular, computed, or dynamic)
`groupParent`	The parent group of this group. A value of 0 means that there is no parent group.

Use the following information to know more about structures.

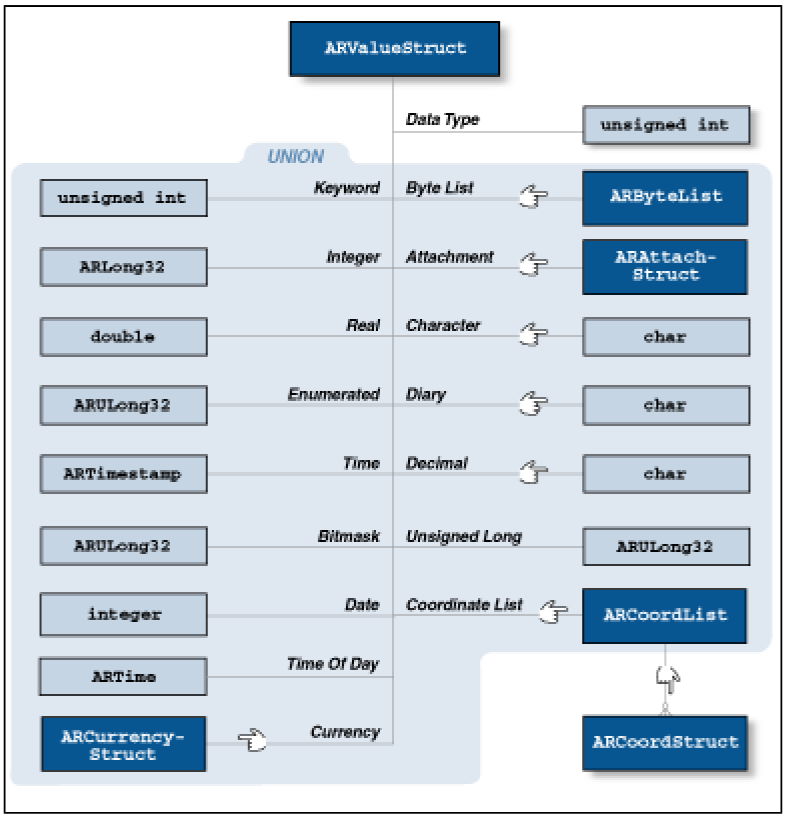
- Representing values with structures
- Representing qualifications with structures
- Schemas and structures
- Fields and structures
- Entries and structures

5.8.3 Representing values with structures

One of the most frequently used data structures is `ARValueStruct`. This structure stores values of any data type.

Many functions, such as `ARCreateField` and `ARGetField`, take parameters of type `ARValueStruct` directly. Other functions use parameters that contain `ARValueStruct`, such as `ARQualifierStruct` and `ARFilterAction`.

Structures used to represent any value



`ARValueStruct` has the following elements:

Data Type	Integer that specifies the data type of the passed value (see the following table).
Value	Value to pass (represented by data types or structures appropriate to the type of value).

0	AR_DATA_TYPE_NULL	Specifies NULL value.
1	AR_DATA_TYPE_KEYWORD	Integer that specifies the particular keyword (defined in the ar.h file).
2	AR_DATA_TYPE_INTEGER	32-bit signed integer.
3	AR_DATA_TYPE_REAL	64-bit floating-point value.
4	AR_DATA_TYPE_CHAR	A null-terminated string that requires freeing allocated space. A NULL pointer of this type is equivalent to using AR_DATA_TYPE_NULL .
5	AR_DATA_TYPE_DIARY	Null-terminated string that requires freeing allocated space. A NULL pointer of this type is equivalent to using AR_DATA_TYPE_NULL .
6	AR_DATA_TYPE_ENUM	

		Set of integers (beginning with zero) that represents possible selection values as an indexed list. You must specify a field limit by using ARNameList to define string values for each selection (see Lists and structures).
7	AR_DATA_TYPE_TIME	UNIX-style date/time stamp (number of seconds since midnight January 1, 1970).
8	AR_DATA_TYPE_BITMASK	32-bit unsigned integer in which each bit represents a flag turned on or off.
9	AR_DATA_TYPE_BYTES	List of bytes containing binary data (represented the ARByteList structure).
10	AR_DATA_TYPE_DECIMAL	Fixed-point decimal field. Values must be specified in C locale, for example, 1234.56 .
11	AR_DATA_TYPE_ATTACH	Attachment field.
12	AR_DATA_TYPE_CURRENCY	Currency field. The numeric part of the values must be specified in C locale, for example, 29.99 .
13	AR_DATA_TYPE_DATE	Date field. The value (known as the Chronological Julian Day) is the integer number of days since January 1, 4713 B.C. The highest valid date is January 1, 9999.
14	AR_DATA_TYPE_TIME_OF_DAY	Time of day field. The value is the integer number of seconds since 12:00:00 a.m.
40	AR_DATA_TYPE_ULONG	32-bit unsigned integer.
41	AR_DATA_TYPE_COORDS	List of (*x*, *y*) coordinate pairs.

Note

When a diary value is passed to the `ARCreate` *Object* or `ARSet` *Object* functions, the `ARValueStruct` value is the *additional* text to append to the diary field. When diary values are retrieved or passed to `ARMergeEntry`, the `ARValueStruct` value is a diary-formatted string containing the full diary text. To decode this string into an array of time-stamped entries, use the `ARDecodeDiary` function.

5.8.4 Representing qualifications with structures

Many tasks in BMC Remedy AR System use qualifications. For example, a qualification (or lack thereof) determines which entries to retrieve when creating a query result list (`ARGetListEntry`) or computing entry statistics (`ARGetEntryStatistics`).

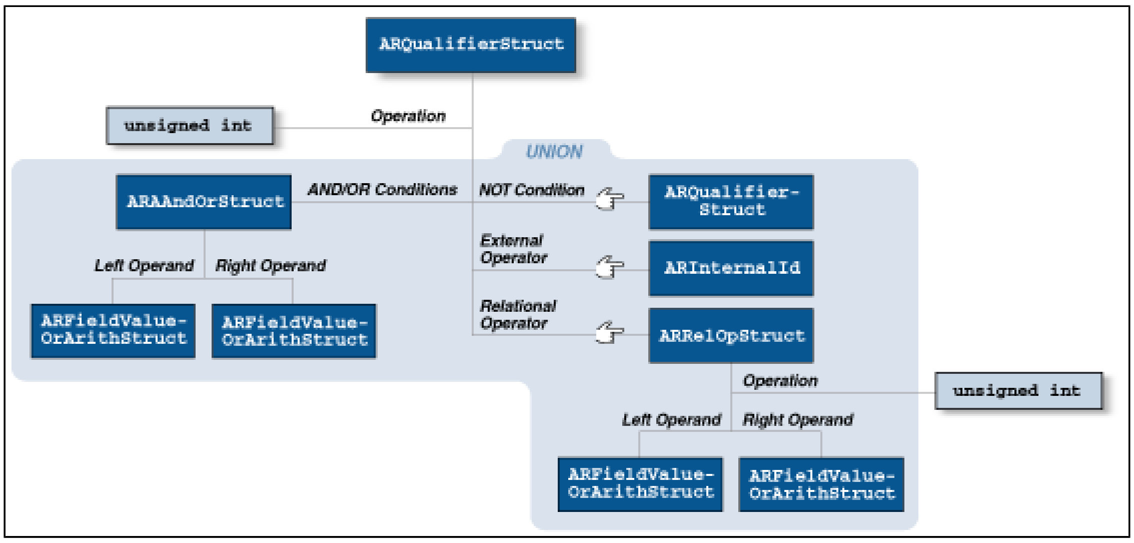
A qualification also determines which active link or filter actions, if any, are executed and which entries an escalation acts on.

Every qualification is composed of a set of zero or more conditions that limit the set of entries retrieved. As the following figure shows, `ARQualifierStruct` uses a tree structure composed of an operation and one or two operands (depending on the type of operation) to represent each individual condition.

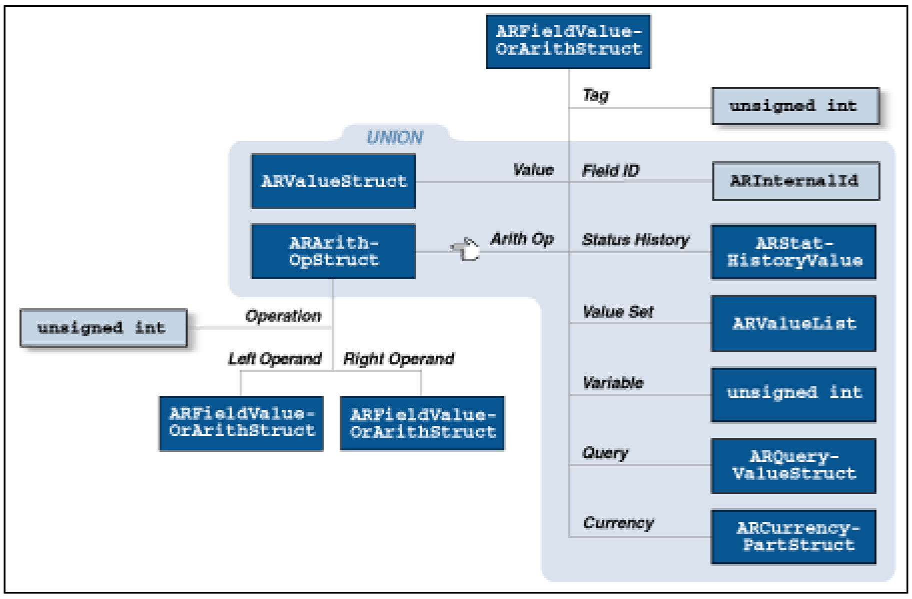
The operands can point to other structures of type `ARQualifierStruct`. This recursion enables you to use a series of nested `ARQualifierStruct` structures to represent any qualification.

To avoid the complexity of storing a qualification in this structure, use ARLoadARQualifierStruct. This function takes the specified qualification string and loads it into an `ARQualifierStruct`. If you are comfortable working with the `ARQualifierStruct` structure or your query is static, you might find that using `ARQualifierStruct` directly yields improved performance.

Structures that represent any qualification--ARQualifierStruct



Structures that represent any qualification--ARFieldValueOrArithStruct



You can also use the following functions to manipulate `ARQualifierStruct`:

- `ARDecodeARQualifierStruct` — Converts information in a serialized qualification string into an `ARQualifierStruct` structure.
- `AREncodeARQualifierStruct` — Converts an `ARQualifierStruct` structure into a serialized qualifier string.

The `ARQualifierStruct` structure has the following elements:

Operation	Integer that specifies the type of *conditional* operation to perform (see the following table).
Operands	Components of the operation (represented by structures appropriate to the type of value).

0	AR_COND_OP_NONE	No qualification.
1	AR_COND_OP_AND	A qualification using the AND operator.
2	AR_COND_OP_OR	A qualification using the OR operator.
3	AR_COND_OP_NOT	A qualification using the NOT operator.
4	AR_COND_OP_REL_OP	A qualification using a relational operator.
5	AR_COND_OP_FROM_FIELD	A qualification located in a field on the schema.

If a query has no qualification criteria, you can either set the operation type to zero (`AR_COND_OP_NONE`) or pass `NULL` for the `qualifier` parameter.

Dynamic qualifications

Dynamic qualifications take part of the qualification condition from a field whose contents can change. The part of the qualification in a field uses the `AR_COND_OP_FROM_FIELD` tag. The value supplied is the ID of the field that contains the qualification.

Qualifications that use conditional operators

Qualifications that use the `AND` or `OR` operators require two operands. Qualifications that use the `NOT` operator require one operand. In both cases, the operands consist of pointers to nested `ARQualifierStruct` structures.

Qualifications that use relational operators

Qualifications that use a relational operator require one operand consisting of a pointer to an `ARRelOpStruct` structure. The `ARRelOpStruct` structure has a tag specifying the operation type and the following operands, which specify the values to compare:

Operation	Integer that specifies the type of *relational* operation to perform (see the following table).
Operands	Components of the operation (represented by `ARFieldValueOrArithStruct` structure).

1	AR_REL_OP_EQUAL	Tests whether the left operand is equal to the right operand.
2	AR_REL_OP_GREATER	Tests whether the left operand is greater than the right operand.
3	AR_REL_OP_GREATER_EQUAL	Tests whether the left operand is greater than or equal to the right operand.
4	AR_REL_OP_LESS	Tests whether the left operand is less than the right operand.
5	AR_REL_OP_LESS_EQUAL	Tests whether the left operand is less than or equal to the right operand.
6	AR_REL_OP_NOT_EQUAL	Tests whether the left operand is not equal to the right operand.
7	AR_REL_OP_LIKE	Tests whether the left operand is LIKE the pattern defined by the right operand.

Defining operands for a relational operation

The `ARFieldValueOrArithStruct` structure represents the values to compare in a relational operation. This structure has the following elements:

Tag	Integer that specifies the type of value (see the following table).
Value	Value to compare (represented by data types or structures appropriate to the type of value).

1	AR_FIELD	Schema field value.
2	AR_VALUE	Constant or keyword value represented by `ARValueStruct` (see Group information and structures).

3	AR_ARITHMETIC	Result value from an arithmetic operation represented by `ARArithOpStruct` (see Defining an arithmetic result value).
4	AR_STAT_HISTORY	Value from the Status-History core field represented by `ARStatHistoryValue` (see Defining a status history value).
5	AR_VALUE_SET	List of values to set.
6	AR_CURRENCY_FLD	Schema currency field value represented by `ARCurrencyPartStruct`.

Defining an arithmetic result value

`ARArithOpStruct` represents the result value from an arithmetic operation. Similar to `ARRelOpStruct` (see Qualifications that use relational operators), this structure has a tag specifying the operation type and two operands specifying the values:

Operation	Integer that specifies the type of *arithmetic* operation to perform (see the following table).
Operands	Components of the operation (represented by `ARFieldValueOrArithStruct`).

1	AR_ARITH_OP_ADD	Add the left and right operands.
2	AR_ARITH_OP_SUBTRACT	Subtract the right operand from the left operand.
3	AR_ARITH_OP_MULTIPLY	Multiply the left and right operands.
4	AR_ARITH_OP_DIVIDE	Divide the left operand by the right operand.
5	AR_ARITH_OP_MODULO	Find the remainder after dividing the left operand by the right operand.
6	AR_ARITH_OP_NEGATE	Change the sign of the right operand (left operand is ignored).

Defining a status history value

The Status History core field is a special selection field that stores user and time stamp information for each of the defined status values. To specify a value for this field, use the `ARStatHistoryValue` structure. This structure has the following elements:

`Index`	Enumerated value that specifies the status value to use in the operation.
`Tag`	Integer that specifies which information about the specified status to use (see the following table).

1	AR_STAT_HISTORY_USER	User who assigned the specified status.
2	AR_STAT_HISTORY_TIME	Time that the status was assigned.

Defining a currency field part

Currency fields consist of several parts that combine to represent a complete currency value. To specify an individual part from this field type or to reference the entire field, use the `ARCurrencyPartStruct` structure. This structure has the following elements:

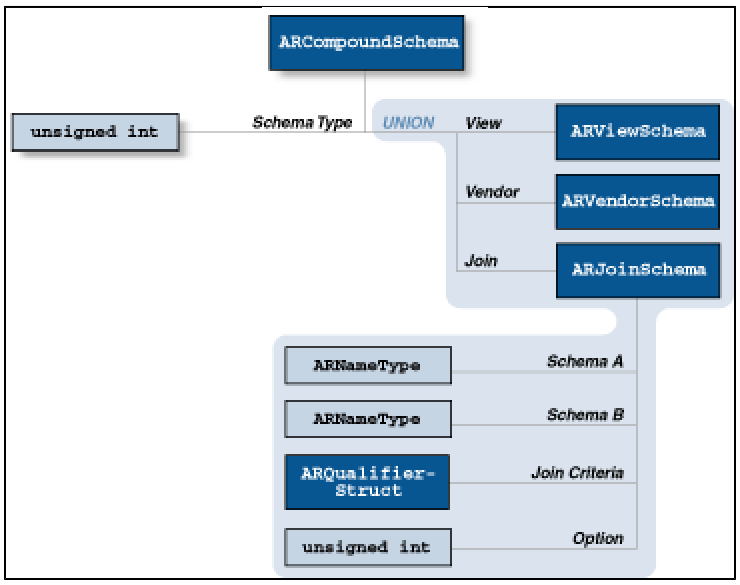
`Field ID`	Internal ID of the currency field (for information about field IDs, see Finding fields by label, name, or ID).
`Part Tag`	Integer that specifies the type of currency part (see the following table).
`Currency Code`	Character string containing the currency code if the Part Tag element is set to `AR_CURRENCY_PART_FUNCTIONAL`.

0	AR_CURRENCY_PART_FIELD	Entire currency field.
1	AR_CURRENCY_PART_VALUE	Numeric currency value.
2	AR_CURRENCY_PART_TYPE	Currency code associated with the currency value.
3	AR_CURRENCY_PART_DATE	Currency conversion date.
4	AR_CURRENCY_PART_FUNCTIONAL	One of the defined functional currencies.

5.8.5 Schemas and structures

The BMC Remedy AR System C API uses the term *schema,* but the BMC Remedy AR System clients refer to schemas as *forms*. Schemas are represented by the `ARCompoundSchema` structure (see the following figure).

Structures used to create join schemas



The `ARCompoundSchema` structure has these elements:

Schema Type	Integer that specifies the type of schema (see the following table).
Join Definition	

	`ARJoinSchema` structure that defines the join schema. It contains the names of the two member schemas (either can be a join schema), the criteria for joining them, and a bitmask of join options (see the following table).
Vendor Definition	`ARVendorSchema` structure that defines the view schema.
View Definition	`ARViewSchema` structure that defines the view schema.

Schema Type values

1	AR_SCHEMA_REGULAR	Base schema.
2	AR_SCHEMA_JOIN	Join schema (has two base schemas).
3	AR_SCHEMA_VIEW	View schema.
4	AR_SCHEMA_DIALOG	Display-only schema (contains only display-only fields).
5	AR_SCHEMA_VENDOR	Vendor schema.

Join Definition options

0	AR_JOIN_OPTION_NONE	No option.
1	AR_JOIN_OPTION_OUTER	Outer join.

5.8.6 Fields and structures

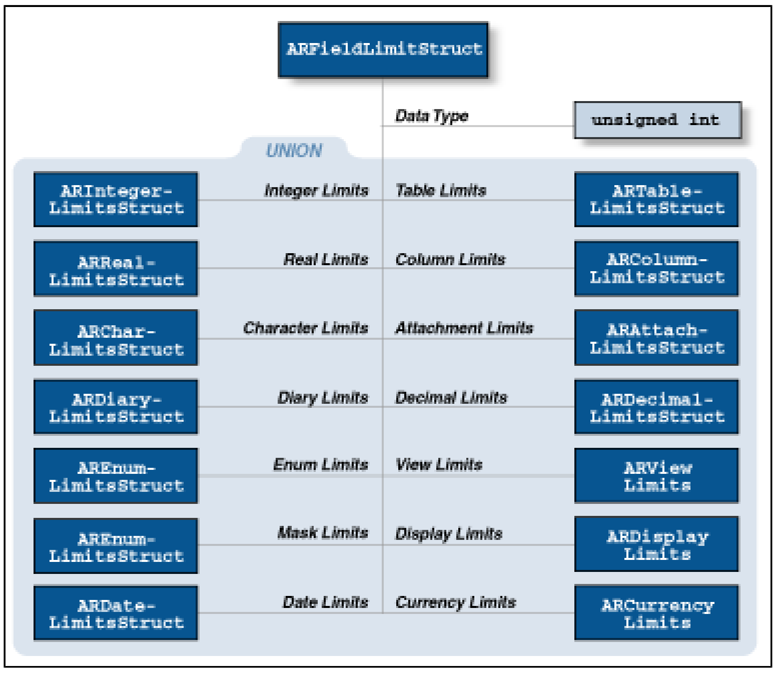
The following section describes some of the structure types related to creating, retrieving, and modifying field definitions. Each of the structures is used as a parameter type for the following functions:

- ARCreateField
- ARGetField
- ARSetField

Defining field limits

The `limit` parameter defines the value limits for a data field. The field's data type determines the type of limits you can specify. This parameter is a pointer to an `ARFieldLimitStruct` structure (see the following figure).

Structures used to define field value boundaries



`ARFieldLimitStruct` enables you to define value limits for data fields of any type, much like `ARValueStruct` enables you to specify values of any data type (see Structures used to represent any value).

The `ARFieldLimitStruct` structure has the following elements:

`Data Type`	Integer that specifies the data type of the field (see the following table).
`Limit`	Limits to assign (represented by structures appropriate to the type of value).

0	AR_FIELD_LIMIT_NONE	Field has no defined limits.
2	AR_DATA_TYPE_INTEGER	Lower and upper range limits, defined by using `ARIntegerLimitsStruct`.
3	AR_DATA_TYPE_REAL	Lower and upper range limits, defined by using `ARRealLimitsStruct`. You can also specify the display precision to use.
4	AR_DATA_TYPE_CHAR	Maximum field length, defined by using `ARCharLimitsStruct`. Specify zero to indicate no maximum. You can also specify a character menu to attach (and whether selecting from the menu appends or overwrites text already in the field), a default query-by-example qualification type, a field character pattern, and whether the field is indexed for full text search (FTS). **Note**

		For more information about storing long character strings, see the Input Length database property in Field Properties.
5	AR_DATA_TYPE_DIARY	Specifies whether the field is indexed for FTS by using `ARDiaryLimitStruct`. **Note** For more information about storing long character strings, see the Input Length database property in Field Properties.
6	AR_DATA_TYPE_ENUM	Values for an enumerated list, defined by using `AREnumListsStruct` (*required*). Only AR_ENUM_STYLE_REGULAR is implemented. The set of integers (beginning with zero) represents possible selection values; the field value limit is defined by using `ARNameList` as a string of values for each selection. For the form of the list structure, see Lists and structures.
10	AR_DATA_TYPE_DECIMAL	Lower and upper range limits, defined by using `ARDecimalLimitsStruct` . You can also specify the precision.
11	AR_DATA_TYPE_ATTACH	Maximum size of the attachment and attachment type (embedded is the only type supported at this time), defined by using `ARAttachLimitsStruct`.
12	AR_DATA_TYPE_CURRENCY	Schema currency field value. Lower and upper range limits, precision, allowable currency definitions, and functional currency definitions, defined by using `ARCurrencyLimitsStruct`.
13	AR_DATA_TYPE_DATE	Lower and upper range limits, defined by using `ARDateLimitsStruct`.
33	AR_DATA_TYPE_TABLE	Number of columns, search qualification, maximum number of rows to retrieve, schema name, and server name of the table, defined by using `ARTableLimitsStruct`.
34	AR_DATA_TYPE_COLUMN	Parent field ID, data field ID (the remote field ID from the data source), data source (the data source for the data field, which can be a data field on a display-only field), and number of characters to display, defined by using `ARColumnLimitsStruct`.
42	AR_DATA_TYPE_VIEW	View field, defined by using `ARViewLimits`.
43	AR_DATA_TYPE_DISPLAY	Values for flashboards for display fields, defined by using `ARDisplayLimits`.

Defining field display properties

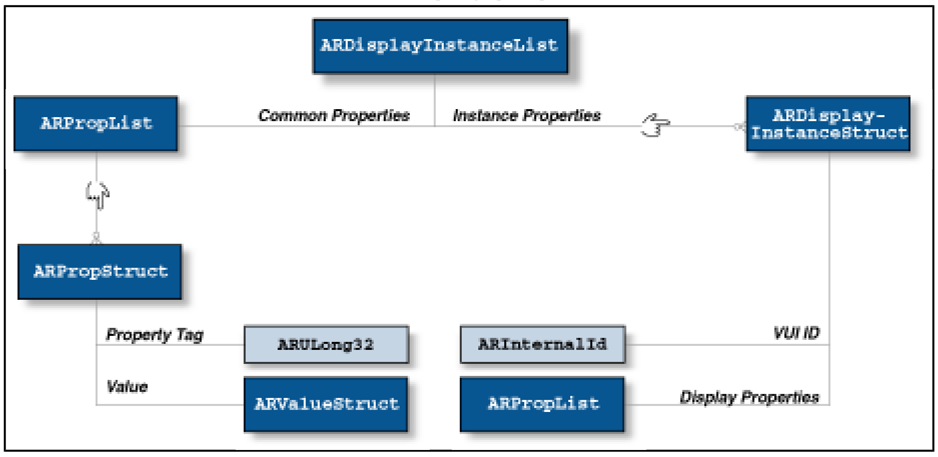
Field display properties (AR_DPROP_*) fall into these categories:

- Those common to all schema views
  These display properties are collected in one ARPropList structure.
- Those specific to a particular schema view
  These display properties are collected in a series of additional ARPropList structures, each linked to a particular view (VUI). These view-specific property lists are represented by zero or more RDisplayInstanceStruct structures.

> **Note**
>
> For information about field *object* properties (AR_OPROP_*), see Server object properties and structures.

Both display property types are represented as lists of zero or more properties in an ARPropList structure. *All* of these structures are collected in an ARDisplayInstanceList structure (see the following figure).

Structures used to define field display properties



The dInstanceList parameter is a pointer to a structure of this type and is used to pass all field display properties when field definitions are created, retrieved, or modified.

The common display properties are represented by an embedded ARPropList structure. This list contains zero or more items, each representing one display property. The ARPropStruct structure has the following elements:

Property Tag	Integer that specifies the display property. For a list of valid integers, see Server object property tags.
Value	Value for the property represented by ARValueStruct (see Group information and structures).

The instance display properties are represented as a series of `ARDisplayInstanceStruct` structures, each of which has the following elements:

`VUI ID`	Internal ID associated with the schema view.
`Display Properties`	Field display properties specific to that view represented by ARPropList .

Each `ARDisplayInstanceStruct` represents a **display instance** for the field. Specifying a schema view in the `ARDisplayInstanceList` structure includes the field in the view, even if you do not define any display properties specific to that view. In this case, the system uses properties defined in the common properties list.

Note

The `ARPropList` structure is also used by the `ARCreateVUI`, `ARGetVUI`, and `ARSetVUI` functions (see BMC Remedy AR System C API functions). The `dPropList` parameter is a pointer to a structure of this type and is used to pass view-type display properties. This data structure is also used for handling object properties (see Server object properties and structures for more information).

Using ARCoordList to specify field size and location

To specify the size and location of certain fields and their components, use the `ARCoordList` data type. These values are scaled by BMC Remedy AR System clients so that fields appear consistent across different user environments.

Bounding boxes and coordinate lists

- **AR_DPROP_BBOX** and **AR_DPROP_COORDS** display properties are `ARCoordList` structures.
- **AR_DPROP_BBOX** properties identify the **bounding box**, or screen boundaries, for screen elements.
- **AR_DPROP_COORDS** identifies the screen location for lines and boxes.

When specifying coordinate values for these properties, remember this:

- The order in which you specify coordinate pairs is important.
- Horizontal and vertical are the only line orientations currently supported by BMC Remedy Developer Studio. Neither client displays diagonal lines for lines or boxes if you specify coordinates that are not on the same *x*- or *y*-axis.

Coordinate value scaling

To minimize display differences across a variety of operating systems and screen resolutions, clients normalize all coordinate values before storing them in the database. They map these values to pixels in the current environment on retrieval. For BMC Remedy Developer Studio to display a field correctly, apply the same translation algorithms when specifying or retrieving coordinate data.

Use this algorithm to normalize coordinate values before storing them:

```
Normalized = (Pixel * AR_FIXED_POINT_PRECISION * staticFontMetric) / (dynamicFontMetric *
platformScale)
```

Use this algorithm to map coordinate values to pixels on retrieval:

```
Pixel = ((Normalized * dynamicFontMetric / staticFontMetric / platformScale) +
(AR_FIXED_POINT_PRECISION / 2)) / AR_FIXED_POINT_PRECISION
```

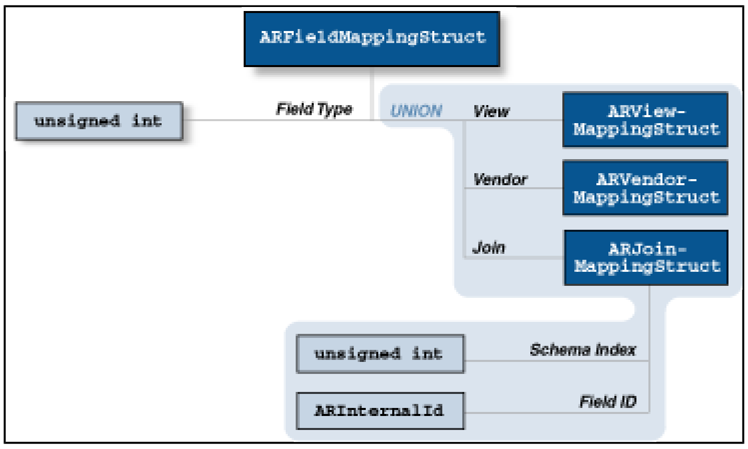
The value for **AR_FIXED_POINT_PRECISION** is 100 on all platforms. The other equation values are platform-specific. This table lists the Windows values:

Equation component	x value	y value
Static Font Metric	9	13
Dynamic Font Metric (using System font style)	(Average character width + maximum character width) / 2	Font height
Platform Scale	1.0	1.0

Mapping fields in schemas

Because schemas (including join, view, and external forms) are logical objects, not actual data tables, you must map each field in a schema to a field in an underlying base schema. This mapping is specified by the `fieldMap` parameter, which is defined as a pointer to an `ARFieldMappingStruct` structure (see the following figure).

Structures used to map join fields to schema fields



The `ARFieldMappingStruct` structure has the following elements:

`Field Type`	Integer that specifies the type of field (see the following table).

`Join Mapping`	`ARJoinMappingStruct` structure that defines the join field mapping. It is described in the next paragraph.
`Vendor Mapping`	`ARVendorMappingStruct` that contains the name of the field in the external database table.
`View Mapping`	`ARViewMappingStruct` that contains the name of the field in the external database table.
`Inheritance Mapping`	`ARInheritanceMappingStruct` is defined for future use. Its use in `ARFieldMappingStruct` is not suppported.

1	AR_FIELD_REGULAR	Field in a base schema.
2	AR_FIELD_JOIN	Field in a join schema.
3	AR_FIELD_VIEW	Field in a view schema.
4	AR_FIELD_VENDOR	Field in a vendor schema.

The `ARJoinMappingStruct` structure has the following components:

`Schema Index`	Integer that specifies the member schema the field maps to (see the following table).
`Field ID`	Internal ID associated with the field.

0	AR_FIELD_MAPPING_PRIMARY	Primary schema in a join.
1	AR_FIELD_MAPPING_SECONDARY	Secondary schema in a join.

If the member schema is also a join schema, you must create fields in all nested join schemas until you can map the field to an underlying base schema.

5.8.7 Entries and structures

The BMC Remedy AR System C API uses the term *entry*, but the BMC Remedy AR System clients refer to entries as *requests*.

The most common operations in BMC Remedy AR System involve creating, retrieving, or updating entries. The following section describes some of the structures used to perform these tasks.

Retrieving entry lists

You can retrieve a list of entries in either of these ways:

- With the fields of each entry as one concatenated string (see ARGetListEntry)
- As field/value pairs (see ARGetListEntryWithFields)

In both cases, the function performs the query specified by the `qualifier` parameter (of type `ARQualifierStruct`) and returns the list of entries that match the search criteria.

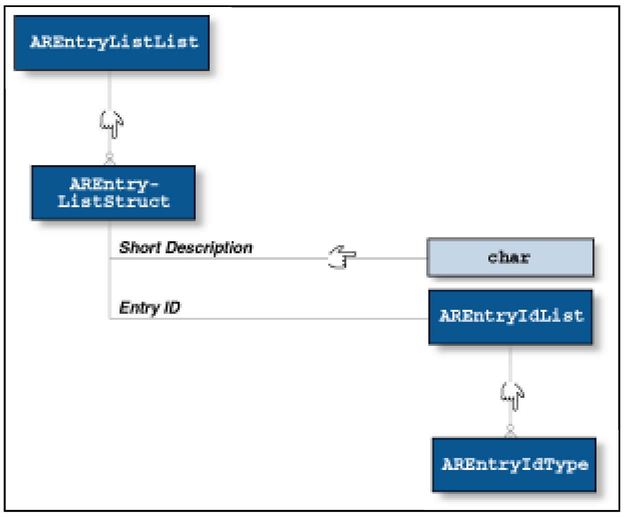
Note

The system identifies entries in join schemas by concatenating the entry IDs from the member schemas. As a result, an entry ID can have one or more values of type `AREntryIdType` and, therefore, is represented by `AREntryIdList`.

Entries returned as concatenated strings

The `ARGetListEntry` function returns an `entryList` output parameter that is a pointer to an `AREntryListList` structure (see the following figure).

Structures used to represent a list of entries as concatenated strings



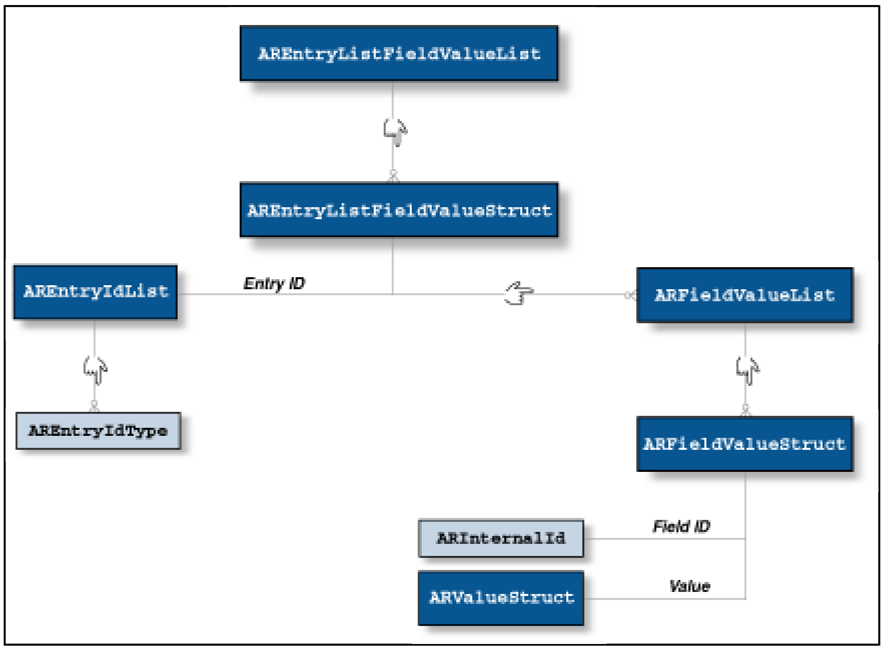
`AREntryListList` has zero or more `AREntryListStruct` items, each representing a matching entry. The `AREntryListStruct` structure has the following elements:

`Short Description`	128-character string containing data concatenated from the fields specified by the `getListFields` parameter.
`Entry ID`	ID associated with the entry.

Entries returned as Field/Value pairs

The `ARGetListEntryWithFields` function returns an `entryList` output parameter that is a pointer to an `AREntryListFieldValueList` structure (see the following figure).

Structures used to represent a list of entries as Field/Value pairs



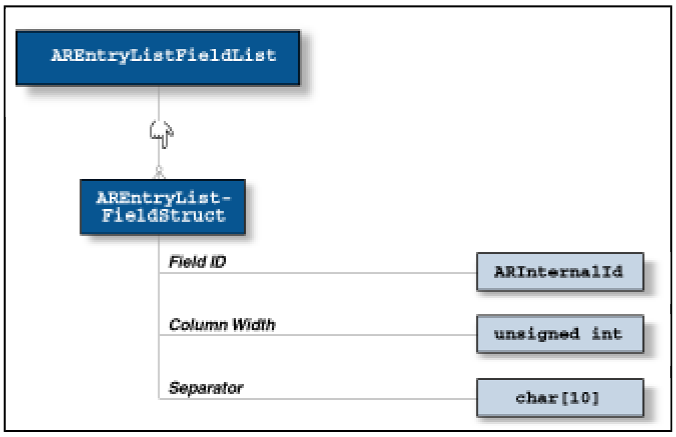
`AREntryListFieldValueList` has zero or more `AREntryListFieldValueStruct` items, each representing a matching entry. The `AREntryListFieldValueStruct` structure has an entry ID and a list of `ARFieldValueStruct` structures, each representing a field in the entry. An `ARFieldValueStruct` structure has the following elements:

`Field ID`	Internal ID associated with the field.
`Value`	Value for the field represented by ARValueStruct (see Group information).

Specifying fields to retrieve

The `getListFields` parameter of both `ARGetListEntry` and `ARGetListEntryWithFields` identifies the fields to return with each entry. It also defines the column width and column separator for each field, which are used by `ARGetListEntry` but ignored by `ARGetListEntryWithFields`. This parameter is a pointer to an `AREntryListFieldList` structure (see the following figure).

Structures used to define column fields in an entry list



Each `AREntryListFieldStruct` item specifies a particular field and associated display information:

`Field ID`	Internal ID associated with the field.
`Column Width`	Integer that specifies the number of characters to display for the field (`ARGetListEntry` only). For `ARGetListEntryWithFields` , set this value to a number greater than 0 .
`Separator`	Characters to display after the field data (`ARGetListEntry` only). The separator can contain from zero to 10 characters. For `ARGetListEntryWithFields` , set this value to one blank space.

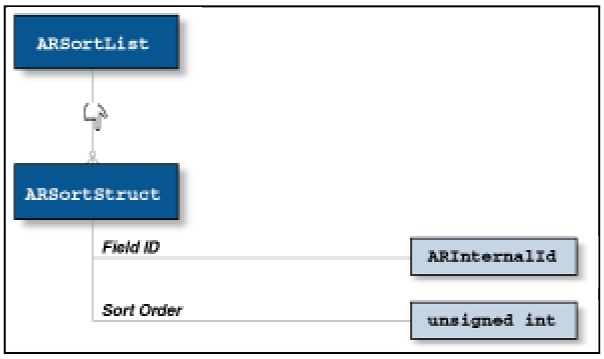
Note

For `ARGetListEntry` , the combined length of all specified fields, including separator characters, can be up to 128 bytes (limited by `AR_MAX_SDESC_SIZE`).

Sorting a list of entries

The `sortList` parameter of both `ARGetListEntry` and `ARGetListEntryWithFields` specifies the sort order for a list of entries. This parameter is a pointer to an `ARSortList` structure.

Structures used to define sort fields in an entry list



As this figure shows, `ARSortList` contains zero or more `ARSortStruct` items. Each `ARSortStruct` has the following elements:

`Field ID`	Internal ID associated with the field.
`Sort Order`	Integer that specifies the sort order for the field (see the following table).

1	AR_SORT_ASCENDING	Sort values in field in ascending order.
2	AR_SORT_DESCENDING	Sort values in field in descending order.

The overall order of entries in the list is determined by the value specified for each field and the order of `ARSortStruct` items in the `ARSortList` structure.

Manipulating individual entries

The ARGetListEntry and ARGetListEntryWithFields functions (see BMC Remedy AR System C API functions) return a list of entries that match specified criteria. This list contains the entry IDs for the matching entries but does not contain all field values. To retrieve or modify individual entries, you must pass the entry ID as an input argument to the ARGetEntry or ARSetEntry functions (see BMC Remedy AR System C API functions).

Both of these functions and the ARCreateEntry and ARGetListEntryWithFields functions use the ARFieldValueList structure to pass field values for individual entries. The ARFieldValueList structure in the following figure has zero or more ARFieldValueStruct items. Each of these structures contains a field and its value (a field/value pair):

Field ID	Internal ID associated with the field.
Value	Value for the field represented by ARValueStruct (see Group information). This value must be of the same data type as the field or have the data type set to zero (AR_DATA_TYPE_NULL).

All entry data, whether being submitted, retrieved, or modified, is represented by this structure.

The following is a code fragment that loads data into this structure.

```
#include "arstruct.h"    /* symbolic constants for core field IDs */

int main( int argc, char *argv[] )
{
    ARFieldValueList     fieldList;     /* structure to populate */
    ARFieldValueStruct  *fldStrcp;      /* working pointer       */
        ...
        ...
/***********************************************************/
/*    Populate fieldList with data                            */
/***********************************************************/
fieldList.numItems = 3;
fieldList.fieldValueList=malloc(3*sizeof(ARFieldValueStruct));
fldStrcp = fieldList.fieldValueList;
if( fldStrcp == NULL){
    fprintf(stderr,"Out of memory, malloc failure\n");
    exit(1);
}
fldStrcp[0].fieldId = AR_CORE_SUBMITTER;
fldStrcp[0].value.dataType = AR_DATA_TYPE_CHAR;
fldStrcp[0].value.u.charVal = "Bob";

fldStrcp[1].fieldId = AR_CORE_STATUS;
fldStrcp[1].value.dataType = AR_DATA_TYPE_ENUM;
fldStrcp[1].value.u.enumVal = STATUS_NEW;

fldStrcp[2].fieldId = AR_CORE_SHORT_DESCRIPTION;
fldStrcp[2].value.dataType = AR_DATA_TYPE_CHAR;
fldStrcp[2].value.u.charVal = "Broken computer";
    ...
    ...
}
```

Note

In this example, the `charVal` members for the Submitter and Short Description fields are on the stack. As a result, you cannot use the `FreeARFieldValueList` function to free this structure because the `FreeAR` routines assume that all structure components are in allocated memory (on the heap). For more information, see Freeing allocated memory.

Retrieving multiple entries

The `ARGetMultipleEntries` function returns a list of entries with the specified entry IDs. You can retrieve data for specific fields or all accessible fields. `ARGetMultipleEntries` performs the same action as `ARGetEntry` but is easier and more efficient than retrieving multiple entries one by one.

This function uses the `ARFieldValueListList` structure to return field values for the specified entries. This structure has zero or more `ARFieldValueList` structures (see the following figure).

`ARGetMultipleEntries` uses the `AREntryIdListList` structure to pass entry IDs for the specified entries. This structure has zero or more `AREntryIdList` structures (see the following figure). `ARGetMultipleEntries` can request a maximum of 100 entries but can be used multiple times.

You can also use the `ARGetMultipleEntries` function to check whether the entries in the `AREntryIdListList` exist without transferring any field data, conserving network traffic. The function has an `existList` parameter that returns a pointer to an `ARBooleanList` structure. Each `ARBoolean` flag in that list specifies whether a given entry ID exists in the database.

5.9 Structures for ARGetListEntryWithMultiSchemaFields

`ARGetListEntryWithMultiSchemaFields` performs dynamic joins, which retrieve entries from multiple forms at runtime. For information about this function, see ARGetListEntryWithMultiSchemaFields and Retrieving entries from multiple forms.

These structures support `ARGetListEntryWithMultiSchemaFields`:

- ARMultiSchemaFuncQueryFromList
- ARMultiSchemaFuncQueryFromStruct
- ARMultiSchemaRecursiveFuncQueryStruct
- ARMultiSchemaNestedFuncQueryStruct
- ARMultiSchemaFieldIdList
- ARMultiSchemaFieldIdStruct

- ARMultiSchemaFieldFuncList
- ARMultiSchemaFieldFuncStruct
- ARMultiSchemaQualifierStruct
- ARMultiSchemaFuncQualifierStruct
- ARMultiSchemaAndOrStruct
- ARMultiSchemaFuncAndOrStruct
- ARMultiSchemaRelOpStruct
- ARMultiSchemaFuncRelOpStruct
- ARMultiSchemaFieldValueOrArithStruct
- ARMultiSchemaFieldFuncValueOrArithStruct
- ARMultiSchemaArithOpStruct
- ARMultiSchemaFuncArithOpStruct
- ARMultiSchemaFuncStatHistoryValue
- ARMultiSchemaFuncCurrencyPartStruct
- ARMultiSchemaFuncValueSetQueryStruct
- ARMultiSchemaSortList
- ARMultiSchemaSortStruct
- ARMultiSchemaFieldFuncValueListList
- ARMultiSchemaFieldFuncValueList
- ARMultiSchemaFieldFuncValueStruct

5.9.1 ARMultiSchemaFuncQueryFromList

Represents a list of zero or more items. The items can include one recursive query and multiple forms (schemas). This structure replaces `ARMultiSchemaQueryFromList` used in earlier releases.

```
typedef struct ARMultiSchemaFuncQueryFromList  {
    unsigned int                                 numItems;
    ARMultiSchemaFuncQueryFromStruct             *listPtr;
} ARMultiSchemaFuncQueryFromList;
```

This structure has these elements:

Element	Description
`numItems`	Integer that specifies the number of items in the list.
`listPtr`	Pointer to each item in the list. Items are represented by `ARMultiSchemaQueryFromStruct`.

5.9.2 ARMultiSchemaFuncQueryFromStruct

Represents an item in the `ARMultiSchemaQueryFromList` structure. An item can be a form (schema) or a recursive query. (The nested query pointer is reserved for future use.) This structure replaces `ARMultiSchemaQueryFromStruct` used in earlier releases.

```
typedef struct ARMultiSchemaFuncQueryFromStruct {
   unsigned int                                   type;
   union {
      ARNameType                                  schemaName;
     struct ARMultiSchemaNestedFuncQueryStruct    *nestedQuery;
     struct ARMultiSchemaRecursiveFuncQueryStruct *recursiveQuery;
   } u;
   ARNameType                                     queryFromAlias;
   unsigned int                                   joinType;
   struct ARMultiSchemaQualifierStruct            *joinQual;
} ARMultiSchemaFuncQueryFromStruct;
```

This structure has these elements:

Element	**Description**
`type`	Integer that specifies the type of item: • Form = 0 (`AR_MULTI_SCHEMA_SCHEMA_NAME`) • Recursive query = 2 (`AR_MULTI_SCHEMA_RECURSIVE_QUERY`)

An `ARMultiSchemaQueryFromList` can contain only one object of the recursive query type, and the recursion can be performed on only one form.

Element	**Description**
`schemaName`	Form name. Specify only when the item type element is set to 0.
`nestedQuery`	Reserved for future use.
`recursiveQuery`	Pointer to a recursive query. The query is represented by `ARMultiSchemaRecursiveQueryStruct`. Specify only when the item type element is set to **2**.
`queryFromAlias`	Alternative name for the specified item. An alias is required for recursive queries and self joins. (In self joins, the same form appears more than once in the `queryFromList` parameter, so each occurrence of the form requires a unique alias.)
`joinType`	Integer that specifies the type of join between this item and the *preceding* item in the list: • `0` — `AR_MULTI_SCHEMA_JOIN_INNER` (Inner join) • `1` — `AR_MULTI_SCHEMA_JOIN_LEFT` (Left outer join) • `2` — `AR_MULTI_SCHEMA_JOIN_RIGHT` (Right outer join) **Note** In the `ARMultiSchemaQueryFromList` structure, specify the join type in the second item and each item thereafter to define the join

	with the previous item. Do not specify the join type in the first item in the structure.
`joinQual`	Pointer to the join criteria, which are represented by `ARMultiSchemaQualifierStruct`. **Note** In the `ARMultiSchemaQueryFromList` structure, specify the join qualifier in the second item and each item thereafter to define the join with the previous item. Do not specify the join qualifier in the first item in the list.

5.9.3 ARMultiSchemaRecursiveFuncQueryStruct

Represents a recursive query. This structure replaces `ARMultiSchemaRecursiveQueryStruct` used in earlier releases.

```
typedef struct ARMultiSchemaRecursiveFuncQueryStruct {
    ARNameType                              recursiveSchemaAlias;
    ARMultiSchemaFuncQueryFromList          queryFromList;
    ARMultiSchemaFieldFuncList              getListFieldFuncs;
    ARMultiSchemaFuncQualifierStruct        *startQual;
    ARMultiSchemaFuncQualifierStruct        *recursionQual;
    int                                     levelsToRetrieve;
} ARMultiSchemaRecursiveFuncQueryStruct;
```

This structure has these elements:

Element	Description
`recursiveSchemaAlias`	Alternative name for the form that contains the hierarchical data. Form aliases are defined in the `ARMultiSchemaQueryFromStruct` structure.
`queryFromList`	List of forms to query, including the form with the hierarchical data and any other forms that you need to join with it to get the required data.
`getListFieldFuncs`	The fields to return with each entry. Each element has a new `funcId` member. • To request the field value, specify `AR_MULTI_SCHEMA_FUNC_NONE`. • To request the value of that function applied to the field, specify `AR_MULTI_SCHEMA_FUNC_ {COUNT,SUM,AVG,MIN,MAX}`

`startQual`	Pointer to a qualification that identifies one or more parent entries for the recursive query.
`recursionQual`	Pointer to a qualification that specifies the conditions (parent-child relationship) that a child record must meet to be returned by the recursive query. This relationship is represented by `ARMultiSchemaQualifierStruct`.
`levelsToRetrieve`	Integer that specifies the number of levels in the hierarchy to retrieve. To retrieve one level below the parent, specify `2`. To retrieve two levels below the parent, specify `3`, and so on. To retrieve all entries in the hierarchy, set this element to `0`. **Note** By default, the maximum value of `AR_MAX_RECURSION_LEVEL_DEFAULT` is 25. To change the default, go to the BMC Remedy AR System Administration: Server Information form's Advanced tab, and enter a new default value in the Maximum Depth for Hierarchical Query field. If the `levelsToRetrieve` element is set to `0`, this setting prevents an infinite loop.

5.9.4 ARMultiSchemaNestedFuncQueryStruct

This structure is reserved for future use.

5.9.5 ARMultiSchemaFieldIdList

Represents a list of zero or more fields.

```
typedef struct ARMultiSchemaFieldIdList {
    unsigned int                  numItems;
    ARMultiSchemaFieldIdStruct    *listPtr;
} ARMultiSchemaFieldIdList;
```

This structure has these elements:

Element	Description
`numItems`	Integer that specifies the number of fields in the list. When this is set to `0`, the query returns the fields in each form's Results List Fields property. If no fields are listed in that property, the query returns the form's Short Description field.

listPtr	Pointers to each field in the list. Fields are represented by ARMultiSchemaFieldIdStruct.

5.9.6 ARMultiSchemaFieldIdStruct

Represents a field qualified by a form (schema) alias.

```
typedef  struct ARMultiSchemaFieldIdStruct {
    ARNameType      queryFromAlias;
    ARInternalId    fieldId;
} ARMultiSchemaFieldIdStruct;
```

This structure has these elements:

Element	Description
queryFromAlias	Alias of the form to which the field belongs. (Aliases are defined in the ARMultiSchemaQueryFromStruct structure.) **Note** Because the ARGetListEntryWithMultiSchemaFields function can operate on multiple forms, each field must be qualified with a form name.
fieldId	Internal ID of the specified field (For information about field IDs, see Finding fields by label, name, or ID).

5.9.7 ARMultiSchemaFieldFuncList

Represents a list to allow specifying more than one field and aggregation function. Used with ARMultiSchemaFieldFuncStruct.

```
typedef struct ARMultiSchemaFieldFuncList {
    unsigned int                   numItems;
    ARMultiSchemaFieldFuncStruct    *listPtr;
} ARMultiSchemaFieldFuncList;
```

This structure has these elements:

Element	Description
numItems	Integer that specifies the number of fields in the list. When this is set to 0, the query returns the fields in each form's Results List Fields property. If no fields are listed in that property, the query returns the form's Short Description field.

`listPtr`	Pointers to each field in the list. Fields are represented by `ARMultiSchemaFieldIdStruct`.

5.9.8 ARMultiSchemaFieldFuncStruct

Identifies the column of interest and indicates the type of aggregation to use. The associated list structure, `ARMultiSchemaFieldFuncList`, allows specifying more than one field and/or aggregation function.

```
typedef struct ARMultiSchemaFieldFuncStruct { /* New */
    ARNameType                queryFromAlias;
    ARInternalId              fieldId;
    Int                       funcId;
} ARMultiSchemaFieldFuncStruct;
```

This structure has these elements:

Element	Description
`queryFromAlias`	Alias of the form to which the field belongs. (Aliases are defined in the `ARMultiSchemaQueryFromStruct` structure.) **Note** Because the `ARGetListEntryWithMultiSchemaFields` function can operate on multiple forms, each field must be qualified with a form name.
`fieldId`	Internal ID of the specified field (For information about field IDs, see Finding fields by label, name, or ID).

`funcId`	Type of aggregation to use. The possible values for the `funcId` member are:
	
	`#define AR_MULTI_SCHEMA_FUNC_NONE         0`
	`#define AR_MULTI_SCHEMA_FUNC_MAX          1`
	`#define AR_MULTI_SCHEMA_FUNC_MIN          2`
	`#define AR_MULTI_SCHEMA_FUNC_AVG          3`
	`#define AR_MULTI_SCHEMA_FUNC_SUM          4`
	`#define AR_MULTI_SCHEMA_FUNC_COUNT        5`

5.9.9 ARMultiSchemaQualifierStruct

Represents a qualification.

```
typedef struct ARMultiSchemaQualifierStruct {
    unsigned int                           operation;
    union {
        ARMultiSchemaAndOrStruct               andor;
        struct ARMultiSchemaQualifierStruct    *notQual;
        struct ARMultiSchemaRelOpStruct        *relOp;
        ARMultiSchemaFieldIdStruct             fieldId;
    } u;
} ARMultiSchemaQualifierStruct;
```

This structure has these elements:

Element	Description
`operation`	Integer that specifies the type of conditional operation to perform: • `0` — `AR_COND_OP_NONE` (No qualification) • `1` — `AR_COND_OP_AND` (Qualification uses the AND operator) • `2` — `AR_COND_OP_OR` (Qualification uses the OR operator) • `3` — `AR_COND_OP_NOT` (Qualification uses the NOT operator) • `4` — `AR_COND_OP_REL_OP` (Qualification uses a relational operator) • `5` — `AR_COND_OP_FROM_FIELD` (Qualification is in a field on the form)
`andor`	Holds information for an AND or OR operation.
`notQual`	Pointer to an `ARMultiSchemaQualifierStruct` structure.
`relOp`	Pointer to `ARMultiSchemaRelOpStruct` structure.
`fieldId`	ID of the field that the qualification operates on. **Note** If used in join criteria, this field cannot be a diary field or a long character field.

5.9.10 ARMultiSchemaFuncQualifierStruct

Represents a qualification, and allows you to use aggregates when constructing a **having** clause in the SQL.

The definition for `ARMultiSchemaFuncQualifierStruct` is:

```
typedef struct ARMultiSchemaFuncQualifierStruct {
    unsigned int                              operation;
    union {
      ARMultiSchemaFuncAndOrStruct            andor;
      struct ARMultiSchemaFuncQualifierStruct *notQual;
      struct ARMultiSchemaFuncRelOpStruct     *relOp;
      ARMultiSchemaFieldFuncStruct            fieldFunc;
```

```
   } u;
} ARMultiSchemaFuncQualifierStruct;
```

This structure has these elements:

Element	Description
`operation`	Integer that specifies the type of conditional operation to perform: • `0` — `AR_COND_OP_NONE` (No qualification) • `1` — `AR_COND_OP_AND` (Qualification uses the AND operator) • `2` — `AR_COND_OP_OR` (Qualification uses the OR operator) • `3` — `AR_COND_OP_NOT` (Qualification uses the NOT operator) • `4` — `AR_COND_OP_REL_OP` (Qualification uses a relational operator) • `5` — `AR_COND_OP_FROM_FIELD` (Qualification is in a field on the form)
`andor`	Holds information for an AND or OR operation.
`notQual`	Pointer to an `ARMultiSchemaQualifierStruct` structure.
`relOp`	Pointer to `ARMultiSchemaRelOpStruct` structure.
`fieldFunc`	ID of the field that the qualification operates on, and allows you to specify the aggregate function to use with the **HAVING** clause. If used in join criteria, this field cannot be a diary field, a long character field, a currency field, or a status history field.

5.9.11 ARMultiSchemaAndOrStruct

Represents operands for AND or OR operators.

```
typedef struct ARMultiSchemaAndOrStruct
{
    struct ARMultiSchemaQualifierStruct    *operandLeft;
    struct ARMultiSchemaQualifierStruct    *operandRight;
} ARMultiSchemaAndOrStruct;
```

This structure has these elements:

Element	Description
`operandLeft`	Pointer to the left component of the operation (represented by `ARMultiSchemaQualifierStruct`).
`operandRight`	Pointer to the right component of the operation (represented by `ARMultiSchemaQualifierStruct`).

5.9.12 ARMultiSchemaFuncAndOrStruct

Represents operands for AND or OR operators, and allows you to specify the aggregate function to use, if any.

```
typedef struct ARMultiSchemaFuncAndOrStruct
{
    struct ARMultiSchemaFuncQualifierStruct     *operandLeft;
    struct ARMultiSchemaFuncQualifierStruct     *operandRight;
} ARMultiSchemaFuncAndOrStruct;
```

This structure has these elements:

Element	Description
`operandLeft`	Pointer to the left component of the operation (represented by `ARMultiSchemaQualifierStruct`).
`operandRight`	Pointer to the right component of the operation (represented by `ARMultiSchemaQualifierStruct`).

5.9.13 ARMultiSchemaRelOpStruct

Represents relational operations.

```
typedef struct ARMultiSchemaRelOpStruct {
    unsigned int                            operation;
    ARMultiSchemaFieldValueOrArithStruct    operandLeft;
    ARMultiSchemaFieldValueOrArithStruct    operandRight;
} ARMultiSchemaRelOpStruct;
```

This structure has these elements:

Element	Description
`operation`	Integer that specifies the type of relational operation to perform: • `1` — `AR_REL_OP_EQUAL` (Tests whether the left operand is equal to the right operand.) • `2` — `AR_REL_OP_GREATER` (Tests whether the left operand is greater than the right operand.) • `3` — `AR_REL_OP_GREATER_EQUAL` (Tests whether the left operand is greater than or equal to the right operand.) • `4` — `AR_REL_OP_LESS` (Tests whether the left operand is less than the right operand.) • `5` — `AR_REL_OP_LESS_EQUAL` (Tests whether the left operand is less than or equal to the right operand.)

	• 6 — `AR_REL_OP_NOT_EQUAL` (Tests whether the left operand is not equal to the right operand.) • 7 — `AR_REL_OP_LIKE` (Tests whether the left operand is LIKE the pattern defined by the right operand.) • 8 — `AR_REL_OP_IN` (Tests whether the specified value is in the value set. The value set can be a set of static strings or a subquery.) • 9 — `AR_REL_OP_NOT_IN` (Tests whether the specified value is *not* in the value set. The value set can be a set of static strings or a subquery.)
`operandLeft`	Holds information for the left operand.
`operandRight`	Holds information for the right operand.

5.9.14 ARMultiSchemaFuncRelOpStruct

Represents relational operations, and allows you to specify the aggregate function to use with a **HAVING** clause.

```
typedef struct ARMultiSchemaFuncRelOpStruct {
    unsigned int                                 operation;
    ARMultiSchemaFieldFuncValueOrArithStruct     operandLeft;
    ARMultiSchemaFieldFuncValueOrArithStruct     operandRight;
} ARMultiSchemaFuncRelOpStruct;
```

This structure has these elements:

Element	Description
`operation`	Integer that specifies the type of relational operation to perform: • 1 — `AR_REL_OP_EQUAL` (Tests whether the left operand is equal to the right operand.) • 2 — `AR_REL_OP_GREATER` (Tests whether the left operand is greater than the right operand.) • 3 — `AR_REL_OP_GREATER_EQUAL` (Tests whether the left operand is greater than or equal to the right operand.) • 4 — `AR_REL_OP_LESS` (Tests whether the left operand is less than the right operand.) • 5 — `AR_REL_OP_LESS_EQUAL` (Tests whether the left operand is less than or equal to the right operand.) • 6 — `AR_REL_OP_NOT_EQUAL` (Tests whether the left operand is not equal to the right operand.) • 7 — `AR_REL_OP_LIKE` (Tests whether the left operand is LIKE the pattern defined by the right operand.)

	• 8 — `AR_REL_OP_IN` (Tests whether the specified value is in the value set. The value set can be a set of static strings or a subquery.) • 9 — `AR_REL_OP_NOT_IN` (Tests whether the specified value is *not* in the value set. The value set can be a set of static strings or a subquery.)
`operandLeft`	Holds information for the left operand.
`operandRight`	Holds information for the right operand.

5.9.15 ARMultiSchemaFieldValueOrArithStruct

Represents the values to compare in a relational operation.

```
typedef struct ARMultiSchemaFieldValueOrArithStruct {
    unsigned int                                   tag;
    union {
        size_t                                     noval_;
        ARMultiSchemaFieldIdStruct                 fieldId;
        ARValueStruct                              value;
        struct ARMultiSchemaArithOpStruct          *arithOp;
        ARMultiSchemaStatHistoryValue              statHistory;
        ARValueList                                valueSet;
        ARMultiSchemaCurrencyPartStruct            *currencyField;
        struct ARMultiSchemaValueSetQueryStruct    *valueSetQuery;
    } u;
```

This structure has these elements:

Element	Description
`tag`	Integer that specifies the value type: • 1 — `AR_FIELD` (Form field value) • 2 — `AR_VALUE` (Constant or keyword value represented by `ARValueStruct`) • 3 — `AR_ARITHMETIC` (Result value from an arithmetic operation represented by `ARMultiSchemaArithOpStruct`) • 4 — `AR_STAT_HISTORY` (Value from the Status-History core field represented by `ARMultiSchemaStatHistoryValue`) • 5 — `AR_VALUE_SET` (List of values to set for the IN or NOT IN operator; the value set must be a set of static strings) • 6 — `AR_CURRENCY_FLD` (Form currency field value represented by `ARMultiSchemaCurrencyPartStruct`) • 7 — `AR_SUB_QUERY` (Subquery used for IN and NOT IN operators)
`noval_`	Element that specifies no value.
`fieldId`	

	ID of a field referenced in the qualifier. When using this structure with the `ARMultiSchemaRecursiveQueryStruct` structure's recursion qualification (`recursionQual`), use the string `"parent"` in the form name (`queryFromAlias`) to identify the parent field ID.
`value`	Pointer to an `ARValueStruct` structure.
`arithOp`	Pointer to an `ARMultiSchemaArithOpStruct` structure.
`statHistory`	Pointer to an `ARMultiSchemaStatHistoryValue` structure.
`valueSet`	A set of values for the IN or NOT IN operator.
`currencyField`	Pointer to an `ARMultiSchemaCurrencyPartStruct` structure.
`valueSetQuery`	Pointer to the IN or NOT IN subquery. The subquery is represented by `ARMultiSchemaValueSetQueryStruct`.

5.9.16 ARMultiSchemaFieldFuncValueOrArithStruct

Represents the values to compare in a relational operation, and allows you to specify the aggregate function to use, if any.

```
typedef struct ARMultiSchemaFieldFuncValueOrArithStruct {
    unsigned int                                  tag;
    union {
      size_t                                      noval_;
      ARMultiSchemaFieldFuncStruct                fieldFunc;
      ARValueStruct                               value;
      struct ARMultiSchemaFuncArithOpStruct       *arithOp;
      ARMultiSchemaStatHistoryValue               statHistory;
      ARValueList                                 valueSet;
      ARMultiSchemaCurrencyPartStruct             *currencyField;
      struct ARMultiSchemaFuncValueSetQueryStruct  *valueSetQuery;
  } ARMultiSchemaFieldFuncValueOrArithStruct;
```

This structure has these elements:

Element	Description
`tag`	Integer that specifies the value type: • 1 — `AR_FIELD` (Form field value) • 2 — `AR_VALUE` (Constant or keyword value represented by `ARValueStruct`) • 3 — `AR_ARITHMETIC` (Result value from an arithmetic operation represented by `ARMultiSchemaArithOpStruct`) • 4 — `AR_STAT_HISTORY` (Value from the Status-History core field represented by `ARMultiSchemaStatHistoryValue`)

	• 5 — AR_VALUE_SET (List of values to set for the IN or NOT IN operator; the value set must be a set of static strings) • 6 — AR_CURRENCY_FLD (Form currency field value represented by ARMultiSchemaCurrencyPartStruct) • 7 — AR_SUB_QUERY (Subquery used for IN and NOT IN operators)
noval_	Element that specifies no value.
fieldFunc	ID of the field that the qualification operates on, and allows you to specify the aggregate function to use with the **HAVING** clause. When using this structure with the ARMultiSchemaRecursiveFuncQueryStruct structure's recursion qualification (recursionQual), use the string "parent" in the form name (queryFromAlias) to identify the parent field ID.
value	Pointer to an ARValueStruct structure.
arithOp	Pointer to an ARMultiSchemaArithOpStruct structure.
statHistory	Pointer to an ARMultiSchemaStatHistoryValue structure.
valueSet	A set of values for the IN or NOT IN operator.
currencyField	Pointer to an ARMultiSchemaCurrencyPartStruct structure.
valueSetQuery	Pointer to the IN or NOT IN subquery. The subquery is represented by ARMultiSchemaValueSetQueryStruct.

5.9.17 ARMultiSchemaArithOpStruct

Represents the result value from an arithmetic operation.

```
typedef struct ARMultiSchemaArithOpStruct {
    unsigned int                            operation;
    ARMultiSchemaFieldValueOrArithStruct    operandLeft;
    ARMultiSchemaFieldValueOrArithStruct    operandRight;
} ARMultiSchemaArithOpStruct;
```

This structure has these elements:

Element	Description
operation	Integer that specifies the type of relational operation to perform: • 1 — AR_ARITH_OP_ADD (Adds the left and right operands) • 2 — AR_ARITH_OP_SUBTRACT (Subtracts the right operand from the left operand) • 3 — AR_ARITH_OP_MULTIPLY (Multiplies the left and right operands) • 4 — AR_ARITH_OP_DIVIDE (Divides the left operand by the right operand)

	• 5 — `AR_ARITH_OP_MODULO` (Finds the remainder after dividing the left operand by the right operand) • 6 — `AR_ARITH_OP_NEGATE` (Changes the sign of the right operand; ignores the left operand)
`operandLeft`	Holds information for the left operand.
`operandRight`	Holds information for the right operand.

5.9.18 ARMultiSchemaFuncArithOpStruct

Represents the result value from an arithmetic operation, and allows you to specify the aggregate function to use, if any.

```
typedef struct ARMultiSchemaFuncArithOpStruct {
    unsigned int                                 operation;
    ARMultiSchemaFieldFuncValueOrArithStruct     operandLeft;
    ARMultiSchemaFieldFuncValueOrArithStruct     operandRight;
} ARMultiSchemaFuncArithOpStruct;
```

This structure has these elements:

Element	Description
`operation`	Integer that specifies the type of relational operation to perform: • 1 — `AR_ARITH_OP_ADD` (Adds the left and right operands) • 2 — `AR_ARITH_OP_SUBTRACT` (Subtracts the right operand from the left operand) • 3 — `AR_ARITH_OP_MULTIPLY` (Multiplies the left and right operands) • 4 — `AR_ARITH_OP_DIVIDE` (Divides the left operand by the right operand) • 5 — `AR_ARITH_OP_MODULO` (Finds the remainder after dividing the left operand by the right operand) • 6 — `AR_ARITH_OP_NEGATE` (Changes the sign of the right operand; ignores the left operand)
`operandLeft`	Holds information for the left operand.
`operandRight`	Holds information for the right operand.

5.9.19 ARMultiSchemaFuncStatHistoryValue

Represents a status history value and allows you to specify the aggregate function, if any.

```
typedef struct ARMultiSchemaFuncStatHistoryValue   {
    ARNameType          queryFromAlias;
    unsigned long       enumVal;
    unsigned int        userOrTime;
    int                 funcId;
} ARMultiSchemaFuncStatHistoryValue;
```

This structure has these elements:

Element	Description
`queryFromAlias`	Alias of the form from which to get the status history. (Aliases are defined in the `ARMultiSchemaQueryFromStruct` structure.)
`enumVal`	Enumerated value that specifies the particular status value to use in the operation.
`userOrTime`	Integer that indicates which information about the status to use: • `1` — `AR_STAT_HISTORY_USER` (User who assigned the specified status) • `2` — `AR_STAT_HISTORY_TIME` (Time that the status was assigned)
`funcId`	A status history field in **WHERE** and **HAVING** qualifiers. In a WHERE qualifier, the `funcId` must be `AR_MULTI_SCHEMA_FUNC_NONE`.

5.9.20 ARMultiSchemaFuncCurrencyPartStruct

Represents part of a currency field and allows you to specify the aggregate function to use, if any.

```
typedef struct ARMultiSchemaFuncCurrencyPartStruct {
    ARMultiSchemaFieldFuncStruct   fieldFunc;
    unsigned int                   partTag;
    ARCurrencyCodeType             currencyCode;
} ARMultiSchemaFuncCurrencyPartStruct;
```

This structure has these elements:

Element	Description
`fieldFunc`	Represents the schema ID, field ID, and function ID of currency fields in **WHERE** and **HAVING** qualifiers. In a **WHERE** qualifier, the function ID must be `AR_MULTI_SCHEMA_FUNC_NONE`.
`partTag`	Integer that specifies the currency part: • `0` — `AR_CURRENCY_PART_FIELD` (Entire currency field) • `1` — `AR_CURRENCY_PART_VALUE` (Numeric currency value) • `2` — `AR_CURRENCY_PART_TYPE` (Currency code associated with the currency value)

	• 3 — `AR_CURRENCY_PART_DATE` (Currency conversion date) • 4 — `AR_CURRENCY_PART_FUNCTIONAL` (One of the defined functional currencies)
`currencyCode`	Character string that contains the currency code if the part tag element is set to `AR_CURRENCY_PART_FUNCTIONAL`.

5.9.21 ARMultiSchemaFuncValueSetQueryStruct

Represents a subquery for the IN or NOT IN operators. Each IN and NOT IN operation can be performed on only one field. This structure replaces `ARMultiSchemaValueSetQueryStruct` used in previous versions.

```
typedef struct ARMultiSchemaFuncValueSetQueryStruct {
    ARMultiSchemaFuncQueryFromList              queryFromList;
    ARMultiSchemaFieldFuncStruct                fieldFunc;
    ARMultiSchemaFuncQualifierStruct            *qualifier;
} ARMultiSchemaFuncValueSetQueryStruct;
```

This structure has these elements:

Element	Description
`queryFromList`	List of items for the primary query to search. Do not include recursive queries in this list.
`fieldFunc`	Internal ID of the field whose value must match a value in the value set, and allows you to specify the aggregate function to use on the field, if any.
`qualifier`	Pointer to the qualifier for the subquery that retrieves the value set. The qualifier is represented by `ARMultiSchemaQualifierStruct`.

5.9.22 ARMultiSchemaSortList

Represents the sort order for a list of entries.

```
typedef struct ARMultiSchemaSortList {
    int                        numItems;
    ARMultiSchemaSortStruct    *listPtr;
} ARMultiSchemaSortList;
```

This structure has these elements:

Element	Description
`numItems`	Integer that specifies the number of items in the list.

`listPtr`	Pointers to each item in the list. Items are represented by `ARMultiSchemaSortStruct`.

5.9.23 ARMultiSchemaSortStruct

Represents a field in the sort order list.

```
typedef struct ARMultiSchemaSortStruct {
    ARMultiSchemaFieldIdStruct    fieldId;
    unsigned int                  sortOrder;
} ARMultiSchemaSortStruct;
```

This structure has these elements:

Element	**Description**
`fieldId`	Form name/field ID pair. The pair is represented by `ARMultiSchemaFieldIdStruct`.
`sortOrder`	Integer that indicates the sort order for the field: • 1 — `AR_SORT_ASCENDING` (Sort values in the field in ascending order) • 2 — `AR_SORT_DESCENDING` (Sort values in the field in descending order)

5.9.24 ARMultiSchemaFieldFuncValueListList

Contains a list of `ARMultiSchemaFieldFuncValueList` structures. Each of these represents one row in the result set returned by the database. (This is analogous to using `ARGetListEntryWithFields`, except that in this case the return values do not contain entry IDs.) The address for this structure is passed to `ARGetListEntryWithMultiSchemaFields`.

```
typedef struct ARMultiSchemaFieldFuncValueListList {
    unsigned int                         numItems;
    ARMultiSchemaFieldFuncValueList      *listPtr;
} ARMultiSchemaFieldFuncValueListList;
```

This structure has these elements:

Element	**Description**
`numItems`	Integer that specifies the number of lists in the list.
`listPtr`	Pointers to each list in the list. Lists are represented by `ARMultiSchemaFieldFuncValueList`.

5.9.25 ARMultiSchemaFieldFuncValueList

Contains a list of zero or more `ARMultiSchemaFieldFuncValueStruct` structures. This list is returned in the order specified by the list of fields being selected (the `getListFields` parameter of `ARGetListEntryWithMultiSchemaFields`).

```
typedef struct ARMultiSchemaFieldFuncValueList {
    unsigned int                       numItems;
    ARMultiSchemaFieldFuncValueStruct  *listPtr;
} ARMultiSchemaFieldFuncValueList;
```

This structure has these elements:

Element	Description
`numItems`	Integer that specifies the number of field values in the list.
`listPtr`	Pointers to each field/value pair in the list.

5.9.26 ARMultiSchemaFieldFuncValueStruct

Represents one field or function value returned by the database. The `fieldId` member contains the form query alias, field ID, and/or function ID, and the `value` member contains the field or function value. (This is again analogous to `ARGetListEntryWithFields`, except that here the return value must be identified by the query alias, the field ID, and the function ID.).

```
typedef struct ARMultiSchemaFieldFuncValueStruct {
    ARMultiSchemaFieldFuncStruct      fieldId;
    ARValueStruct                     value;
} ARMultiSchemaFieldFuncValueStruct;
```

This structure has these elements:

Element	Description
`fieldId`	The form query alias, field ID, and/or function ID.
`value`	The field or function value.

5.10 Filters, escalations, and active links and structures

Use the following topics to understand the various structure types related to creating, retrieving, and modifying filters, escalations, and active links definitions.

- Server object actions and structures
- Set Fields action and structures

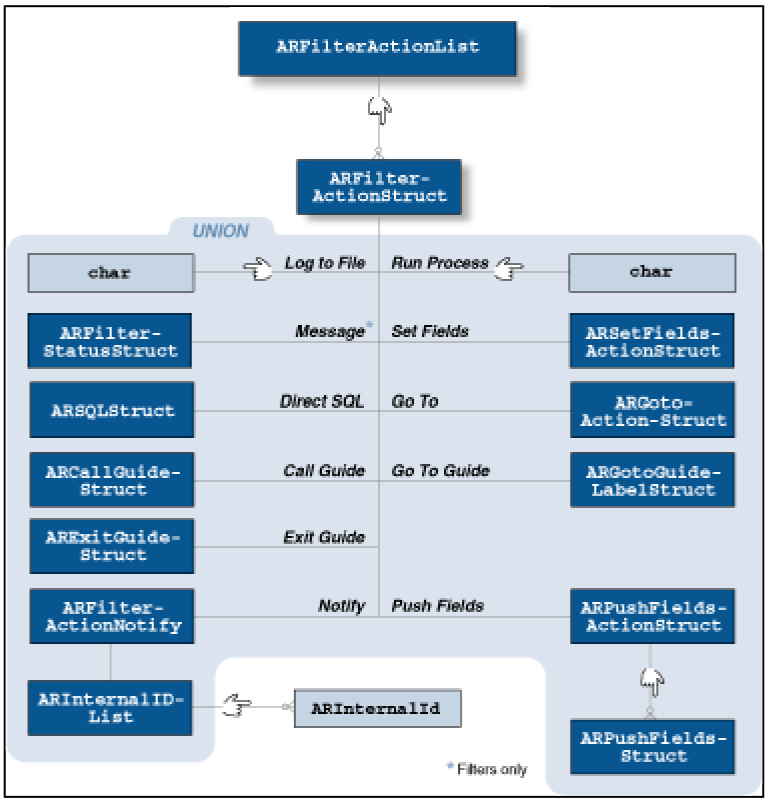
- Push Fields action and structures
- Automation action and structures

5.10.1 Server object actions and structures

The `ARCreate Object`, `ARGet Object`, and `ARSet Object` functions for filters, escalations, and active links have an `actionList` parameter that defines the set of one or more actions performed for entries that match the associated qualification. Because the actions available for filters and escalations are different from those for active links, the actions associated with these objects are represented by different data structures.

For filters and escalations, the `actionList` parameter is a pointer to an `ARFilterActionList` structure (see the following figure).

Structures used to define filter or escalation actions



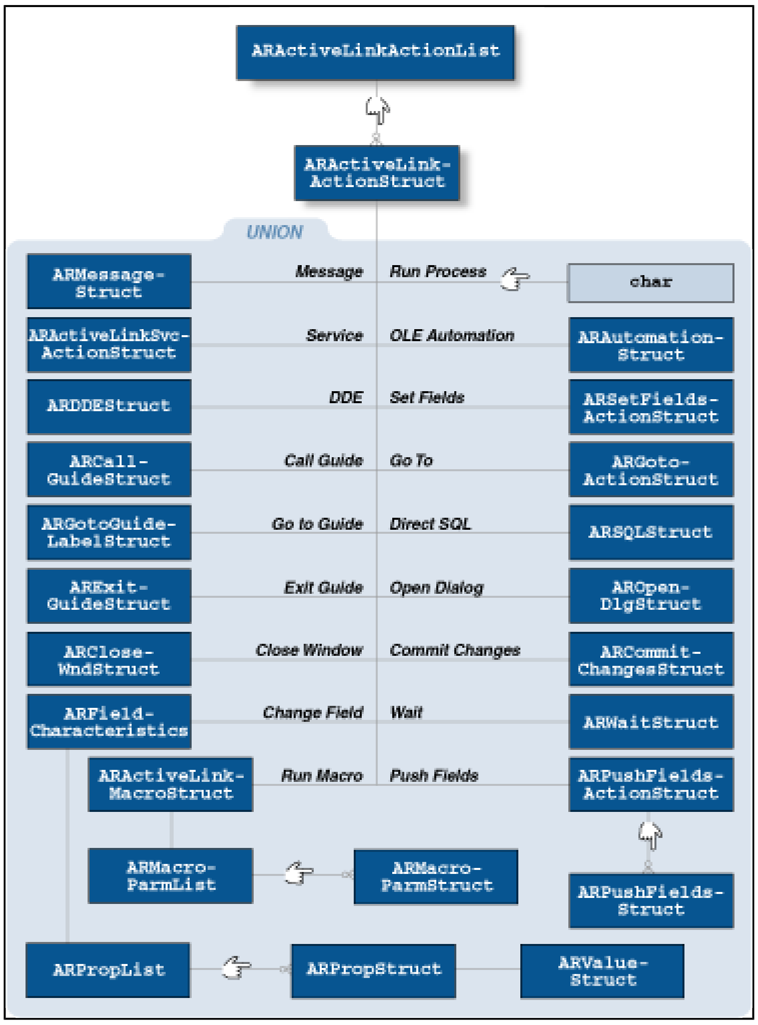
`ARFilterActionList` contains zero or more `ARFilterActionStruct` items. Each **ARFilterActionStruct** represents one filter or escalation action and has the following elements:

`Action Type`	Integer that specifies the type of action (see the following table).
`Action Definition`	Action to perform (represented by data types or structures appropriate to the type of action).

1	AR_FILTER_ACTION_NOTIFY	Send a notification to the specified user.
2	AR_FILTER_ACTION_MESSAGE	Display the specified message (filters only).
3	AR_FILTER_ACTION_LOG	Write the specified text to a log file.
4	AR_FILTER_ACTION_FIELDS	Set values for the specified fields.
5	AR_FILTER_ACTION_PROCESS	Execute the specified command.
6	AR_FILTER_ACTION_FIELDP	Transfer, or push, data from a field in the current request to a field in a request of another schema on the same server.
7	AR_FILTER_ACTION_SQL	Perform the specified SQL statements.
8	AR_FILTER_ACTION_GOTOACTION	Go to the filter with the specified execution order.
9	AR_FILTER_ACTION_CALLGUIDE	Perform the specified guide.
10	AR_FILTER_ACTION_EXITGUIDE	Exit the current guide or all guides.
11	AR_FILTER_ACTION_GOTOGUIDELABEL	Go to the filter with the specified guide label in the current guide.
12	AR_FILTER_ACTION_SERVICE	Return a result without making a database change.

For active links, the `actionList` parameter is a pointer to an `ARActiveLinkActionList` structure (see the following figure).

Structures used to define active link actions



The `ARActiveLinkActionList` structure contains zero or more `ARActiveLinkActionStruct` items. Like `ARFilterActionStruct`, each `ARActiveLinkActionStruct` represents one active link action and has the following elements:

`Action Type`	Integer that specifies the type of action (see the following table).
`Action Definition`	Action to perform (represented by data types or structures appropriate to the type of action).

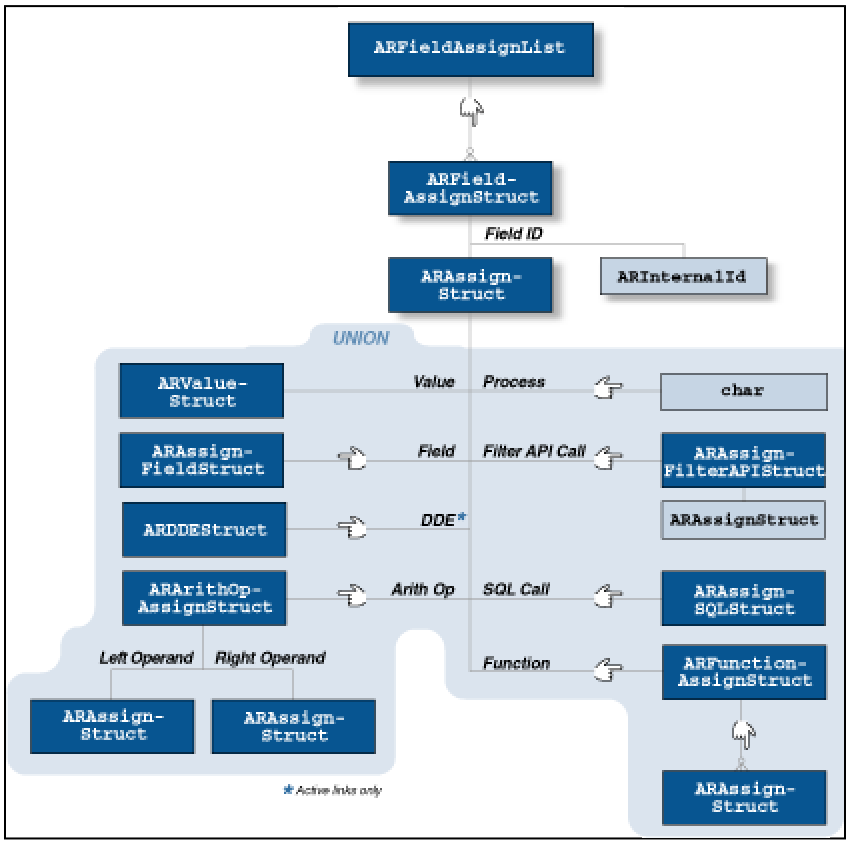
1	AR_ACTIVE_LINK_ACTION_MACRO	Execute the specified macro.
2	AR_ACTIVE_LINK_ACTION_FIELDS	Set values for the specified fields.
3	AR_ACTIVE_LINK_ACTION_PROCESS	Execute the specified command.
4	AR_ACTIVE_LINK_ACTION_MESSAGE	Display the specified message.
5	AR_ACTIVE_LINK_ACTION_SET_CHAR	Set characteristics for the specified field.
6	AR_ACTIVE_LINK_ACTION_DDE	Perform the specified DDE action.
7	AR_ACTIVE_LINK_ACTION_FIELDP	

		Transfer, or push, data from a field in the current request to a field in a request of another schema on the same server.
8	AR_ACTIVE_LINK_ACTION_SQL	Perform the specified SQL statements.
9	AR_ACTIVE_LINK_ACTION_AUTO	Perform the specified OLE automation.
10	AR_ACTIVE_LINK_ACTION_OPENDLG	Open a schema as a modal dialog.
11	AR_ACTIVE_LINK_ACTION_COMMITC	Commit changes from a dialog to the parent schema window.
12	AR_ACTIVE_LINK_ACTION_CLOSEWND	Close a dialog or schema window.
13	AR_ACTIVE_LINK_ACTION_CALLGUIDE	Perform the specified guide.
14	AR_ACTIVE_LINK_ACTION_EXITGUIDE	Exit the current guide or all guides.
15	AR_ACTIVE_LINK_ACTION_GOTOGUIDELABEL	Go to the active link with the specified guide label in the current guide.
16	AR_ACTIVE_LINK_ACTION_WAIT	Make the current guide wait for user input.
17	AR_ACTIVE_LINK_ACTION_GOTOACTION	Go to the active link with the specified execution order.
18	AR_ACTIVE_LINK_ACTION_SERVICE	Return a result without making a database change.

5.10.2 Set Fields action and structures

The Set Fields action is one of the most complex of the filter, escalation, or active link actions. The `ARFieldAssignList` structure (see the following figure) is a list of `ARFieldAssignStruct` items, each identifying a field/value pair to update. Each `ARFieldAssignStruct` structure contains the internal ID of the field to update and the value to assign, specified in `ARAssignStruct`, which is one of the more complex structures in the API.

Structures used to define Set Fields action



The `ARAssignStruct` structure contains the following elements:

`Assign Type`	Integer that specifies the type of value to assign (see the following table).
`Value`	Value to assign (represented by data types or structures appropriate to the type of value).

1	AR_ASSIGN_TYPE_VALUE	Constant or keyword value represented by the structure (see Group information). The data type of the value must match the data type of the specified field.
2	AR_ASSIGN_TYPE_FIELD	Schema field value represented by the structure (see Assigning a schema field value).
3	AR_ASSIGN_TYPE_PROCESS	Output value from successful execution of an operating system process or command. The system generates an error if the process return code is not zero. This option is unavailable for active links on Windows.
4	AR_ASSIGN_TYPE_ARITH	Result value from an arithmetic operation represented by the structure (see Assigning an arithmetic result value).
5	AR_ASSIGN_TYPE_FUNCTION	Return value from a function represented by the structure (see Assigning a function return value).
6	AR_ASSIGN_TYPE_DDE	

		Result value from a DDE request to another application represented by the structure (see Assigning a DDE result value). This option is available only for active links on Windows clients.
7	AR_ASSIGN_TYPE_SQL	Result value from an SQL command represented by the structure (see Assigning an SQL result value).
8	AR_ASSIGN_TYPE_FILTER_API	Result value from a Filter API command represented by the structure (see Assigning a filter API result value).

Assigning a schema field value

The `ARAssignStruct` structure contains an `ARAssignFieldStruct` structure that identifies a schema field value to assign in a Set Fields or Push Fields action. You can specify a value from any entry in any schema on a particular server.

In a Push Fields action, an independent `ARAssignFieldStruct` structure (see Structures used to define a Push Fields action) identifies the schema field set by the operation. You can specify a field from any entry in any schema on a particular server. To set or push all matching field values from one schema to the other, use `AR_LIKE_ID` in the field.

The `ARAssignFieldStruct` structure has the following elements:

`Server`	String that specifies the name of the server where the schema is located. The function retrieves a list of available servers (see BMC Remedy AR System C API functions). For filters and escalations, specify the server where the filter or escalation is located (either by name or by passing). For active links, specify *** to retrieve the value from the current window.
`Schema`	Name of the schema containing the field value to assign. For filters and escalations, specify @ to retrieve or set the value from the current transaction. For active links, specify *** to retrieve or set the value from the current window.
`Search Criteria`	Qualification that identifies the entries to retrieve or set (optional). Specify the search criteria by using `ARQualifierStruct` (see Structures that represent any qualification--ARQualifierStruct and Structures that represent any qualification--ARFieldValueOrArithStruct).
`Tag`	Integer that specifies the type of field value to retrieve or set. The following table lists the valid integers.
`Field ID/Stat History/Currency`	Field containing the value to assign or set. Specify the field by using the field ID, the `ARStatHistoryValue` structure, or the `ARCurrencyPartStruct` structure. Use the field identification to set or push matching field values between two schemas. For more information, see Mapping fields to values.
`No Match Option`	Integer that specifies the action to take if *no* entries match the search criteria. The following table lists the valid integers.

`Multiple Match Option`	Integer that specifies the action to take if *multiple* entries match the search criteria (Set Fields) or if any entry matches the search criteria (Push Fields). The following table lists the valid integers.

Tag values for `ARAssignFieldStruct`

1	AR_FIELD	Value from a regular schema field.
4	AR_STAT_HISTORY	Value from the Status History core field.
6	AR_CURRENCY_FLD	Value from a currency field.

No Match Option values for `ARAssignFieldStruct`

1	AR_NO_MATCH_ERROR	Return an error.
2	AR_NO_MATCH_SET_NULL	Assign (Set Fields action only).
3	AR_NO_MATCH_NO_ACTION	Do nothing (Push Fields action only).
4	AR_NO_MATCH_SUBMIT	Submit a new entry (Push Fields action only).

Multiple-Match Option values for `ARAssignFieldStruct`

1	AR_MULTI_MATCH_ERROR	Return an error
2	AR_MULTI_MATCH_SET_NULL	Assign (Set Fields action only).
3	AR_MULTI_MATCH_USE_FIRST	Assign a value from the first matching entry.
4	AR_MULTI_MATCH_PICKLIST	Display a selection list (Active Links only).
5	AR_MULTI_MATCH_MODIFY_ALL	Modify all matching entries (Push Fields action only).
6	AR_MULTI_MATCH_NO_ACTION	Do nothing (Push Fields action only).
7	AR_MULTI_MATCH_USE_LOCALE	Assign a value from the first matching entry based on the locale (Set Fields action or Active Links only).

Assigning an arithmetic result value

The `ARArithOpAssignStruct` structure defines an arithmetic result value to assign in a Set Fields action. This structure is similar to the `ARArithOpStruct` structure and uses the same set of operation values (see Defining an arithmetic result value). Like `ARArithOpStruct`, `ARArithOpAssignStruct` has a tag specifying the type of arithmetic operation and two operands specifying the values. In this case, however, the operands are represented by `ARAssignStruct` instead of `ARFieldValueOrArithStruct`.

Assigning a function return value

The `ARFunctionAssignStruct` structure specifies a function return value to assign in a Set Fields action. This structure has the following elements:

`Function Type`	Integer that specifies the type of function to perform (see the following table).
`Number of Parameters`	Integer that specifies the number of function input parameters.
`Parameter List`	Pointer to the parameter values to use (represented in a structure).

For more information about the function type values, see Functions.

Function type values for ARFunctionAssignStruct

Number	Function Type	Input	Return	Value
1	AR_FUNCTION_DATE	timestamp	char	Date
2	AR_FUNCTION_TIME	timestamp	char	Time
3	AR_FUNCTION_MONTH	timestamp	long	Month (1-12)
4	AR_FUNCTION_DAY	timestamp	long	Day (1-31)
5	AR_FUNCTION_YEAR	timestamp	long	Year
6	AR_FUNCTION_WEEKDAY	timestamp	long	Weekday (1-7)
7	AR_FUNCTION_HOUR	timestamp	long	Hour (0-23)
8	AR_FUNCTION_MINUTE	timestamp	long	Minute (0-59)
9	AR_FUNCTION_SECOND	timestamp	long	Second (0-59)
10	AR_FUNCTION_TRUNC	real	long	Truncated value
11	AR_FUNCTION_ROUND	real	long	Rounded value
13	AR_FUNCTION_LENGTH	char	long	Length of string in bytes
14	AR_FUNCTION_UPPER	char	char	Uppercase string
15	AR_FUNCTION_LOWER	char	char	Lowercase string
16	AR_FUNCTION_SUBSTR	char, long [, long]	char	Substring from long1 to long2 (inclusive) in bytes
17	AR_FUNCTION_LEFT	char, long	char	Leftmost *n* bytes
18	AR_FUNCTION_RIGHT	char, long	char	Rightmost *n* bytes
19	AR_FUNCTION_LTRIM	char	char	Leading blanks removed
20	AR_FUNCTION_RTRIM	char	char	Trailing blanks removed
21	AR_FUNCTION_LPAD	char, long, char	char	char1 left-padded with char2 to *n* bytes
22	AR_FUNCTION_RPAD	char, long, char	char	char1 right-padded with char2 to *n* bytes

23	AR_FUNCTION_REPLACE	char, char, char	char	char1 with char2 replaced by char3
24	AR_FUNCTION_STRSTR	char, char	int	Position of char2 in char1 in bytes (-1 = not found)
25	AR_FUNCTION_MIN	*x*, *x*, [, *x*] ...	*x*	Minimum value
26	AR_FUNCTION_MAX	*x*, *x*, [, *x*] ...	*x*	Maximum value
27	AR_FUNCTION_COLSUM			Sum a table column
28	AR_FUNCTION_COLCOUNT		int	Number of non-null values in a table column
29	AR_FUNCTION_COLAVG			Average of non-null values in a table column
30	AR_FUNCTION_COLMIN			Minimum of non-null values in a table column
31	AR_FUNCTION_COLMAX			Maximum of non-null values in a table column
32	AR_FUNCTION_DATEADD	char, int, date	date	Number of days, weeks, months, or years to add to date
33	AR_FUNCTION_DATEDIFF	char, date, date	int	Number of days or weeks between the start date and end date
34	AR_FUNCTION_DATENAME	char, date	char	Name of the day or month corresponding to date
35	AR_FUNCTION_DATENUM	char, date	int	Depending on the value of char (year, month, week, day, or weekday), returns the numeric value of the year, month (1 to 12), week (1 to 52), day (1 to 31) or weekday (1 =Sunday, 2 =Monday, and so on)
36	AR_FUNCTION_CURRCONVERT	currency, char, timestamp		Currency values to convert, based on timestamp
37	AR_FUNCTION_CURRSETDATE	currency, timestamp		Date of currency and functional currency values
38	AR_FUNCTION_CURRSETTYPE	currency, char		Type of currency and functional currency values
39	AR_FUNCTION_CURRSETVALUE			

		currency, char		Value of currency and functional currency values
40	AR_FUNCTION_LENGTHC	char	long	Length of string in characters
41	AR_FUNCTION_LEFTC	char, long	char	Leftmost *n* characters
42	AR_FUNCTION_RIGHTC	char, long	char	Rightmost *n* characters
43	AR_FUNCTION_LPADC	char, long, char	char	char1 left-padded with char2 to *n* characters
44	AR_FUNCTION_RPADC	char, long, char	char	char1 right-padded with char2 to *n* characters
45	AR_FUNCTION_STRSTRC	char, char	int	Position of char2 in char1 in characters (-1 = not found)
46	AR_FUNCTION_SUBSTRC	char, long [,long]	char	Substring from long1 to long2 (inclusive) in characters
47	AR_FUNCTION_ENCRYPT	(plaintext, key)	cypher-text	Encrypted value of a text string (plaintext), using the encryption key (key)
48	AR_FUNCTION_DECRYPT	(cypher-text, key)	plaintext	Unencrypted text value of the encrypted text (cyphertext), using the encryption key (key)
49	AR_FUNCTION_HOVER	int	char	Depending on the type of object hovered over (table column, table, attachment pool) returns a column value of a row, a row value, or an attachment file name; for all other fields, returns an empty string; the input value is a field ID
50	AR_FUNCTION_TEMPLATE	char, char, char [, char, char ...]	char	Number of arguments is variable; the first argument is the template name; the second argument is a parameter name; the third argument is the corresponding parameter value; subsequent arguments are parameter name-value pairs; therefore, this function always requires an odd number of arguments

Note

If necessary, the system automatically converts all input parameters to the correct data type.

Assigning a DDE result value

The `ARDDEStruct` structure defines a DDE result value to assign in a **SetFields** action. This option is available only for active links on Windows clients. The `ARDDEStruct` structure has the following elements:

`Service Name`	String that specifies the service to use.
`Topic`	String that specifies the topic to use.
`Item`	String that specifies the item to retrieve.
`Action`	Integer that specifies the type of DDE action. Specify AR_DDE_REQUEST for this item.
`Path`	String that specifies the path to the application.
`Command`	String that specifies the command to execute. Specify NULL for this item.

Assigning an SQL result value

The `ARAssignSQLStruct` structure specifies an SQL result value to assign in a **SetFields** action. Use this option to assign a value from any SQL database. The `ARAssignSQLStruct` structure has the following elements:

`Server`	String that specifies the name of the server where the database is located. The ARGetListServer function retrieves a list of available servers (see BMC Remedy AR System C API functions).
`SQL Command`	String that specifies the SQL query to execute.
`Value Index`	Integer that specifies which of the selected columns contains the value to assign.
`No Match Option`	Integer that specifies the action to take if *no* rows match the selection criteria. The following table lists the valid integers.
`Multiple Match Option`	Integer that specifies the action to take if *multiple* rows match the selection criteria. The following table lists the valid integers.

Assigning a filter API result value

The `ARAssignFilterApiStruct` structure specifies the name of the plug-in, the input values for the operation, and a pointer to the output values. The `ARAssignFilterApiStruct` structure has the following elements:

`Service Name`	String that specifies the name of the plug-in.
`Input Values`	List of values provided as input to the BMC Remedy AR System filter (AR Filter) API plug-in.
`Return Index`	Index into the returned list of values used in the set field assignment.

Note

You can assign AR Filter API result values for escalation Set Field actions, but you cannot assign AR Filter API result values for active link Set Field actions.

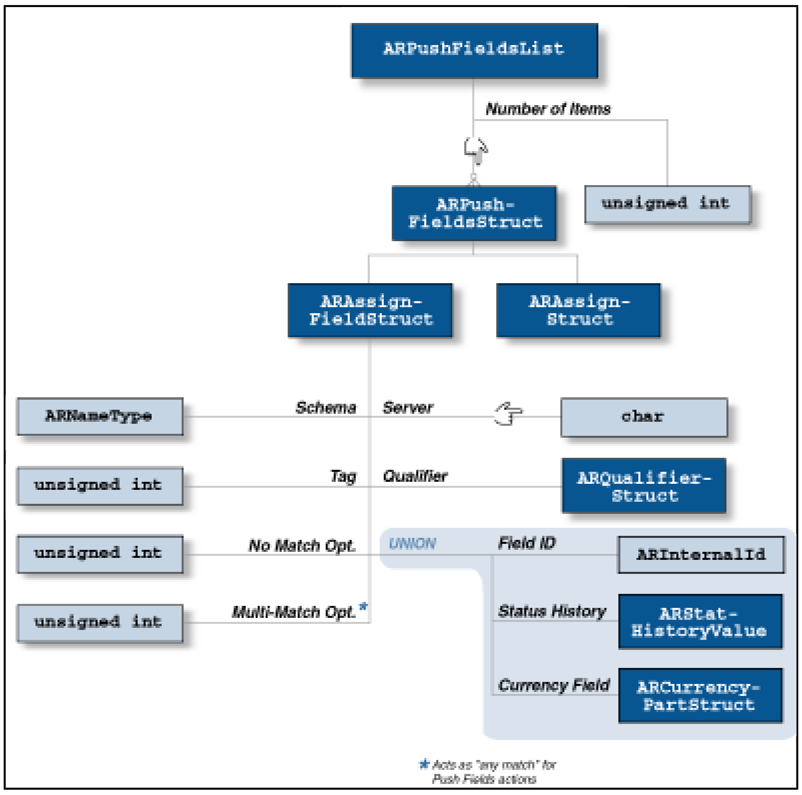
5.10.3 Push Fields action and structures

While the Set Fields action sets the value of a field in the current schema, the Push Fields action modifies or submits entries in any schema. When Push Fields is used in an active link, the schema can be on any server. In the following figure, the `ARPushFieldsList` structure is a list of `ARPushFieldsStruct` items. Each item contains an `ARAssignFieldStruct` structure that specifies the field to update and an `ARAssignStruct` structure that specifies the value to push to the field. These data structures are also used in Set Fields actions.

Note

To force a Push Fields action to perform a submit, set the Qualifier to `AR_COND_OP_NONE`, the No Match option to `AR_NO_MATCH_SUBMIT`, and the Multiple Match option to `AR_MULTI_MATCH_NO_ACTION`.

Structures used to define a Push Fields action

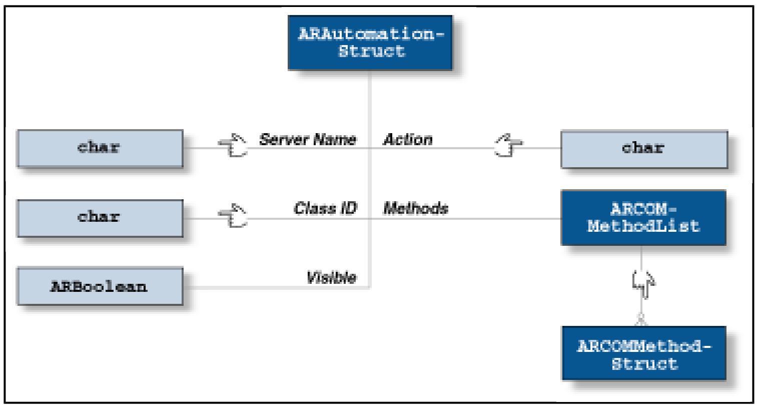


5.10.4 Automation action and structures

The Automation active link action enables a Windows client to control Automation servers on the same computer, accessing the data and methods that those servers make available.

The `ARAutomationStruct` structure (see the following figure) defines an Automation action.

Structures used to define automation action

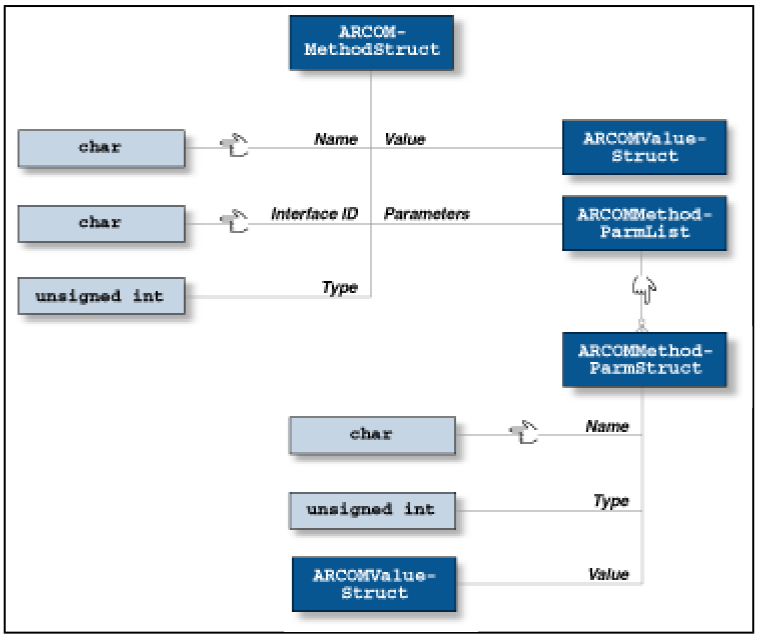


The `ARAutomationStruct` structure has the following elements:

`Server Name`	String that specifies the name of the Automation server, length limited by `AR_MAX_COM_NAME` (64 bytes).
`Class ID`	String that specifies the unique ID assigned to this server in the registry, length limited by `AR_MAX_COM_ID_SIZE` (128 bytes).
`Action`	String that specifies the equation defined by this action, including nested methods and the assignment if any. For example:`$Status$ = MethodA().MethodB(MethodC())` This string is used only for display and is limited by **AR_MAX_AUTOMATION_SIZE** (2000 bytes).
`Visible`	Not used. Specify 0 (zero).
`Methods`	List of methods called by this active link action, specified in an `ARCOMMethodList` structure. List the methods in execution order. For example, if Method B is passed as a parameter to Method A, list Method B first so that its result value is available when Method A is called.

Each method in an `ARCOMMethodList` structure uses the `ARCOMMethodStruct` structure (see the following figure).

Structures used to define OLE automation method



The `ARCOMMethodStruct` structure has the following elements:

`Name`	Method name, length limited by `AR_MAX_COM_METHOD_NAME` (128 bytes).
`Interface ID`	ID of the interface that owns the method, length limited by `AR_MAX_COM_ID_SIZE` (128 bytes).
`Type`	Type of value returned by the method, if any. Possible values are all OLE variant types.
`Value`	

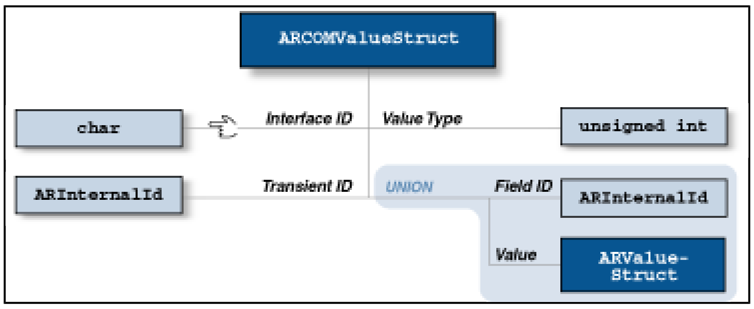
	BMC Remedy AR System field to set to the method's return value, if any. This is specified in an `ARCOMValueStruct` structure (see the following figure).
`Parameters`	List of parameters passed to this method, specified in an `ARCOMMethodParmList` structure.

Each parameter in an `ARCOMMethodParmList` structure is specified in an `ARCOMMethodParmStruct` structure (see the following figure), which has the following elements:

`Name`	Name of the parameter.
`Type`	Data type of the parameter. Possible values are all OLE variant types.
`Value`	Value of the parameter, specified in an `ARCOMValueStruct` structure.

The value returned by a method and the value passed in a parameter are each specified in the `ARCOMValueStruct` structure (see the following figure).

Structures used to define OLE method and parameter values



The `ARCOMValueStruct` structure has the following elements:

`Interface ID`	One of these values: • When this structure defines the return value of a method, specify the ID of the interface returned by the method. • When this structure defines the value of a parameter that is an OLE interface, specify the ID of that interface. • Otherwise, specify `NULL`.
`Transient ID`	Link between a method that takes another method as a parameter to the method used as the parameter. For example, Method A takes Method B as a parameter. Use the same integer for the `ARCOMValueStruct` structure that defines the return value for Method B as you do for the structure that defines the parameter value of Method A.
`Value Type`	Integer that specifies the type of value represented by the structure (see the following table). For Value Type 0, set the other elements of the structure to NULL or 0.

`Field ID/Value`	Union that defines the field ID or value, depending on the Value Type: • Value Type 1--ID of the field to set to the return value of this method or to pass as this parameter, specified in an `ARInternalId` structure. • Value Type 2--Value to pass as this parameter, specified in an `ARValueStruct` structure.

Method return value types for ARCOMValueStruct

0	AR_COM_METHOD_NULL	No value returned for this method.
1	AR_COM_METHOD_FIELDID	Set the specified field to the value returned by this method.

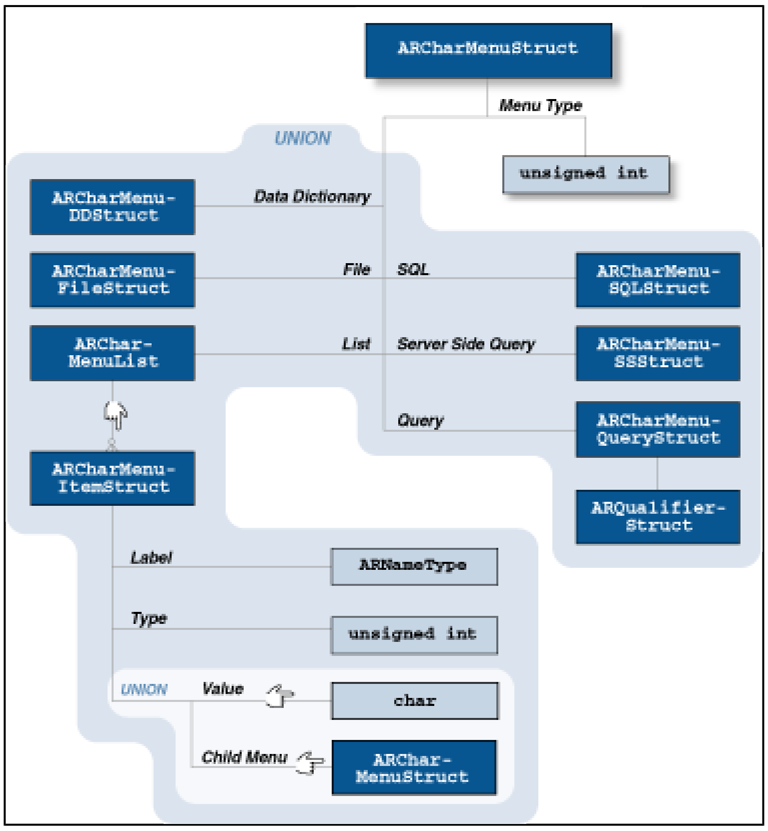
Parameter value types for ARCOMValueStruct

0	AR_COM_PARM_NULL	No value for this parameter.
1	AR_COM_PARM_FIELDID	Pass the specified field as the parameter.
2	AR_COM_PARM_VALUE	Pass the specified value as the parameter.

5.11 Character menus and data structures

Character menu definitions are represented by the `ARCharMenuStruct` structure (see the following figure).

Structures used to define character menus



Most character menu functions have a `menuDefn` parameter that is a pointer to a structure of this type (for example, see ARCreateCharMenu or ARExpandCharMenu).

The `ARCharMenuStruct` structure defines both the content and organization of the menu and has the following elements:

`Menu Type`	Integer that specifies the type of character menu (see the following table).
`Menu Definition`	Menu content (represented by structures appropriate to the type of menu).

1	AR_CHAR_MENU_LIST	Menu items based on definitions in `ARCharMenuList`.
2	AR_CHAR_MENU_QUERY	Menu items based on schema query defined in `ARCharMenuQueryStruct`.
3	AR_CHAR_MENU_FILE	Menu items based on formatted flat file defined in `ARCharMenuFileStruct`.
4	AR_CHAR_MENU_SQL	Menu items based on SQL query defined in `ARCharMenuSQLStruct`.
5	AR_CHAR_MENU_SS	Menu items based on server side query defined in `ARCharMenuSSStruct`.
6	AR_CHAR_MENU_DATA_DICTIONARY	

		Menu items based on data dictionary query defined in `ARCharMenuDDStruct`.

The `ARCharMenuList` structure has zero or more `ARCharMenuItemStruct` items, each representing an individual menu item. In this context, a menu item can be a value (a leaf item) or another menu (a top- or intermediate-level item).

The `ARCharMenuItemStruct` structure has the following elements:

`Label`	Label that identifies the menu item.
`Type`	Integer that specifies the type of menu item (see the following table).
`Menu Definition`	Value associated with the menu item. For leaf-level items, the definition is a string containing the item text. For top- or intermediate-level items, the definition is a pointer to a child menu (represented by `ARCharMenuStruct`).

1	AR_MENU_TYPE_VALUE	Leaf menu item.
2	AR_MENU_TYPE_MENU	Top- or intermediate-level menu item.

5.12 Images and data structures

The `ARGetImage` function returns image contents in an `ARImageDataStruct`:

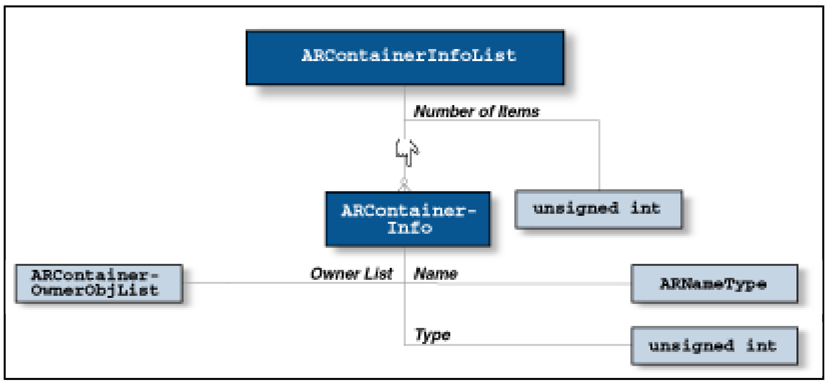
```
typedef struct ARImageDataStruct
{
   unsigned int      numItems;      /* length in bytes     */
   unsigned char     *bytes;        /* not NULL terminated */
} ARImageDataStruct;
} ARImageDataStruct;
typedef struct ARImageDataList
{
   unsigned int      numItems;
   ARImageDataStruct *imageList;
   ARImageDataStruct *imageList;
} ARImageDataList;    /* list of 0 or more image data structs */
```

5.13 Containers and structures

Containers are generic lists of references used to define guides and applications. Use the following information to understand how structures are used with containers.

- Retrieving container lists
- Manipulating individual containers

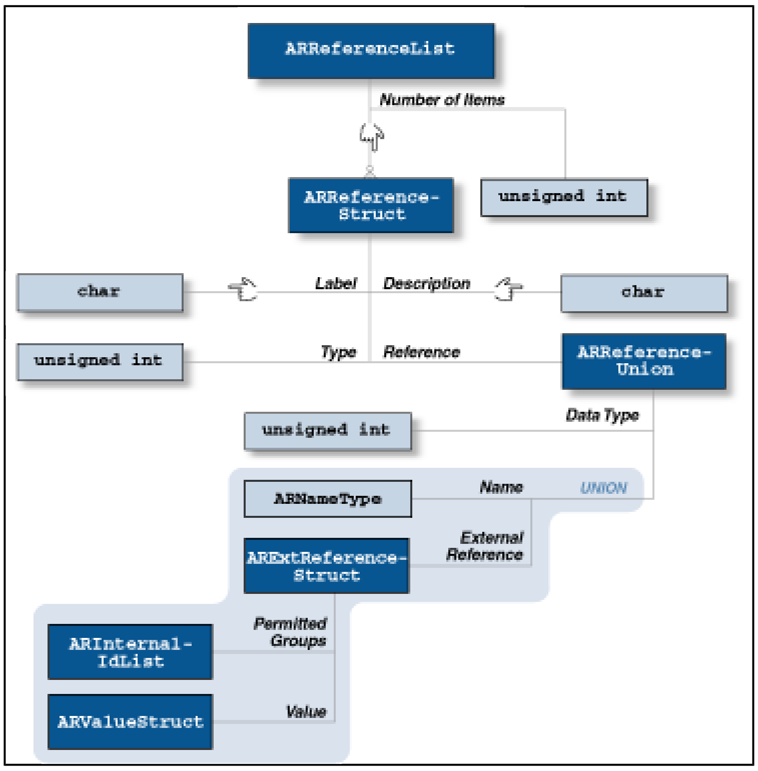
5.13.1 Retrieving container lists



To retrieve entry lists, call the `ARGetListContainer` function. The function returns a list of containers that match the specified type, owning object, and modification time stamp.

The `ARGetListContainer` function uses the `ARContainerInfoList` structure to specify the containers it returns. The `ARContainerInfoList` structure has zero or more `ARContainerInfo` structures (see the following figure).

Structures used to define container lists



The `ARContainerInfoList` structure has the following elements:

`Name`	`ARNameType` structure that specifies the name of the container.

`Type`	Integer that specifies the type of container (see the following table).
`Owner List`	List of structures that limit the containers retrieved, based on the object that owns them.

Container type values for ARContainerInfo

1	ARCON_GUIDE	Guide.
2	ARCON_APP	Application.
3	ARCON_PACK	Packing list.
4	ARCON_FILTER_GUIDE	Filter guide.
5	ARCON_WEBSERVICE	Web service.

Owner type values for ARContainerInfo

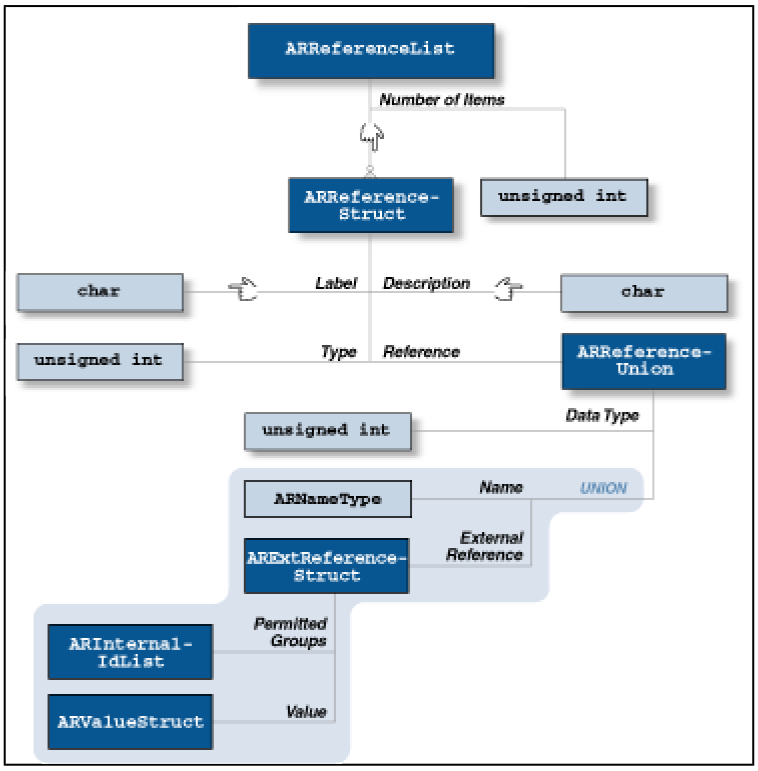
0	ARCONOWNER_NONE	Unowned container (global).
1	ARCONOWNER_ALL	Owned and unowned containers.
2	ARCONOWNER_SCHEMA	Owner object is a schema.

5.13.2 Manipulating individual containers

The ARGetListContainer function returns a list of containers that match specified criteria. The list contains only the names of the matching containers, not their reference lists and other attributes. To retrieve or modify individual containers, pass the name as an input argument to the ARGetContainer or ARSetContainer functions.

Both of these functions and the ARCreateContainer function use the `ARReferenceList` structure to specify the objects referenced by a container. Guides reference active links. Applications reference schemas and other objects, both internal and external. The `ARReferenceList` structure has zero or more `ARReferenceStruct` structures (see the following figure).

Structures used to define container references



The `ARReferenceStruct` structure has the following elements:

`Label`	String that specifies a display-only label for the reference. Its length is limited by `ARMAX_CON_LABEL_LEN` (255 bytes).
`Description`	String that specifies a display-only description for the reference. Its length is limited by `ARMAX_CON_DESCRIPTION_LEN` (2000 bytes).
`Type`	Integer that specifies the type of reference represented by the structure (see the following table). Values above 32767 define external reference types.
`Reference`	Union that defines an internal or external reference, depending on its Data Type, which is either 0 (`ARREF_DATA_ARSREF`) or 1 (`ARREF_DATA_EXTREF`). Internal references (0) are specified in an `ARInternalId` structure. External references (1) are specified in an `ARExtReferenceStruct` structure (see the following figure).

Reference type values for ARReferenceStruct

0	ARREF_NONE	No reference.
1	ARREF_ALL	All reference types.
2	ARREF_SCHEMA	Reference to a schema.
3	ARREF_FILTER	Reference to a filter.
4	ARREF_ESCALATION	Reference to an escalation.

5	ARREF_ACTLINK	Reference to an active link.
6	ARREF_CONTAINER	Reference to a container.
7	ARREF_CHAR_MENU	Reference to a character menu.
32768	ARREF_ICON	Reference to an icon.
32769	ARREF_SMALL_ICON	Reference to a small icon.
32770	ARREF_MAXIMIZE_FORMS	Reference to a Boolean value that specifies whether to maximize the forms of an application.
32771	ARREF_APPLICATION_FORMS	Tag that specifies that the next reference in the list is a schema. Use this in front of each schema reference in an application except the primary schema.
32772	ARREF_ABOUT_BOX_IMAGE	Reference to an About box image.
32773	ARREF_ABOUT_BOX_FORM	Tag that specifies that the About box form is the next reference.
32774	ARREF_NULL_STRING	Reference to a null string.
32775	ARREF_APPLICATION_HELP_EXT	Reference to a string that specifies the help file name extension.
32776	ARREF_APPLICATION_HELP_FILE	Reference to a bytelist that specifies the contents (not the name) of the help file to use with the application.
32777	ARREF_APPLICATION_PRIMARY_FORM	Tag that specifies that the next reference in the list is the primary schema for this application. Use this in front of only one schema reference in an application.
32778	ARREF_APPLICATION_FORM_VUI	Reference to the ID of the schema view to use with the previous schema referenced in the list. Use this after each schema reference in an application.
32779	ARREF_APPLICATION_DISABLE_BEGIN_TASK	Reference to a Boolean value that specifies whether the Begin a Task menu item is disabled for the application. The default is false (not disabled).
32780	ARREF_APPLICATION_HELP_INDEX_EXT	Reference to the application object help file's index file extension.
32781	ARREF_APPLICATION_HELP_INDEX_FILE	Reference to the application object help file's index file.
32782	ARREF_APPLICATION_HELP_FILE_NAME	

		Reference to the application object help file's name without the extension.
32783	ARREF_PACKINGLIST_GUIDE	Packing list reference to a guide.
32784	ARREF_PACKINGLIST_APP	Packing list reference to an application.
32785	ARREF_PACKINGLIST_PACK	Packing list reference to a packing list.
32786	ARREF_GROUP_DATA	Packing list reference to data in the group schema.
32787	ARREF_DISTMAPPING_DATA	Packing list reference to data in the Distributed Mapping schema.
32788	ARREF_APPLICATION_HAS_EXT_HELP	Reference to whether applications use external help (Boolean).
32789	ARREF_APPLICATION_SUPPORT_FILES	Reference to the application object support file's contents.
32792	ARREF_PACKINGLIST_DSOPOOL	Packing lists reference to data in the Distributed Pool schema.
32793	ARREF_PACKINGLIST_FILTER_GUIDE	Packing lists reference to a filter guide.
32794	ARREF_FLASH_BOARD_DEF	Reference to a Flashboard definition.
32795	ARREF_FLASH_DATA_SOURCE_DEF	Reference to a Flashboard data source definition.
32796	ARREF_FLASH_VARIABLE_DEF	Reference to a Flashboard variable definition.
32797	ARREF_WS_PROPERTIES	XML string that refers to miscellaneous web service properties.
32798	ARREF_WS_OPERATION	Reference to web service operation information (XML string consisting of name, type, mapping names, and so on).
32799	ARREF_WS_ARXML_MAPPING	Mapping XML document that describes the relationship between BMC Remedy AR System model and XML schema.
32800	ARREF_WS_WSDL	Reference to WSDL for a web service.
32801	ARREF_PACKINGLIST_WEBSERVICE	Packing lists reference to a web service.
32802	ARREF_WS_PUBLISHING_LOC	Reference to saved URLs for publishing a web service.
32803	ARREF_APPLICATION_HELP_FILE_NAME2	Reference to the application object help file's name without the extension.
32804	ARREF_APPLICATION_HELP_EXT2	Reference to the application object help file's extension.

32805	ARREF_APPLICATION_HELP_FILE2	Reference to the application object help file.
32806	ARREF_APPLICATION_HELP_INDEX_EXT2	Reference to the application object help file's index file extension.
32807	ARREF_APPLICATION_HELP_INDEX_ FILE2	Reference to the application object help file's index file.
32808	ARREF_APPLICATION_HELP_FILE_NAME3	Reference to the application object help file's name without the extension.
32809	ARREF_APPLICATION_HELP_EXT3	Reference to the application object help file's extension.
32810	ARREF_APPLICATION_HELP_FILE3	Reference to the application object help file.
32811	ARREF_APPLICATION_HELP_INDEX_EXT3	Reference to the application object help file's index file extension.
32812	ARREF_APPLICATION_HELP_INDEX_ FILE3	Reference to the application object help file's index file.
32813	ARREF_APPLICATION_HELP_FILE_NAME4	Reference to the application object help file's name without the extension.
32814	ARREF_APPLICATION_HELP_EXT4	Reference to the application object help file's extension.
32815	ARREF_APPLICATION_HELP_FILE4	Reference to the application object help file.
32816	ARREF_APPLICATION_HELP_INDEX_EXT4	Reference to the application object help file's index file extension.
32817	ARREF_APPLICATION_HELP_INDEX_ FILE4	Reference to the application object help file's index file.
32818	ARREF_APPLICATION_HELP_FILE_NAME5	Reference to the application object help file's name, without the extension.
32819	ARREF_APPLICATION_HELP_EXT5	Reference to the application object help file's extension.
32820	ARREF_APPLICATION_HELP_FILE5	Reference to the application object help file.
32821	ARREF_APPLICATION_HELP_INDEX_EXT5	Reference to the application object help file's index file extension.
32822	ARREF_APPLICATION_HELP_INDEX_ FILE5	Reference to the application object help file's index file.
32823	ARREF_APPLICATION_HELP_LABEL	Reference to the application object help file's label.
32824	ARREF_APPLICATION_HELP_LABEL2	

		Reference to the application object help file's label.
32825	ARREF_APPLICATION_HELP_LABEL3	Reference to the application object help file's label.
32826	ARREF_APPLICATION_HELP_LABEL4	Reference to the application object help file's label.
32827	ARREF_APPLICATION_HELP_LABEL5	Reference to the application object help file's label.
32828	ARREF_WS_XML_SCHEMA_LOC	Reference to a web service's XML schema location.
32829	ARREF_ENTRYPOINT_ORDER	Reference to the listing order of the entry point.
32830	ARREF_ENTRYPOINT_START_ACTLINK	Reference to the starting active link for the entry point.
32831	ARREF_APP_AUTOLAYOUT_SS	Reference to style sheet information for auto layout.
32832	ARREF_APP_FORMACTION_FIELDS	Reference to form action fields.
32833	ARREF_ENCAPSULATED_APP_DATA	Reference to the application data identifier.
32835	ARREF_APP_DEFAULT_OBJ_PERMS	Reference to default application object permissions.
32836	ARREF_APP_ADD_FORMACTION_FIELDS	Reference to add form action fields.
32837	ARREF_APP_FORMACTION_RESULTS_ LIST_FIXED_HEADER	Reference to a fixed header property.
32838	ARREF_APP_FORMACTION_PAGE_ PROPERTIES	Reference to a page property.
32839	ARREF_APP_OBJECT_VERSION	Reference to the application object version.
32840	ARREF_APP_PACKING_LISTS	Reference to packing lists in the application.
32841	ARREF_APP_DATA_MERGE_IMP_QUAL	Reference to a list of field IDs for merge qualifications on import.
32842	ARREF_APP_DATA_MERGE_IMP_OPTION	Reference to the data import option for merge on application import.

The `ARExtReferenceStruct` structure (see the above figure) defines external references. This structure has the following elements:

`Permitted Groups`	`ARInternalIdList` structure that specifies the groups that have access to the referenced object.

`Value`	`ARValueStruct` structure that specifies the external reference.

5.14 Overlays and structures

The following structure supports the creation of overlay objects. For more information about overlay objects, see About overlays.

5.14.1 AROverlaidStruct

Represents an object to be overlaid. This structure is used by the `ARCreateOverlay` and `ARCreateOverlayFromObject` functions.

```
typedef struct AROverlaidStruct {
    ARNameType            name;
    ARInternalID          objectId;
    unsigned int          obType;
    ARNameType            schemaName;
    unsigned int          inheritMask;
    unsigned int          extendMask;
 } AROverlaidStruct;
```

This structure has the following elements:

Element	Description
`name`	Name of the object to be overlaid.
`objectId`	(Fields and views only) ID of the field or view object to be overlaid. This value is required.
`obType`	Type of object to be overlaid (active link, active link guide, escalation, filter, filter guide, form, view, field, image, local application, menu, packing list, or web service).
`schemaName`	(Fields and views only) Name of the form with which the object to be overlaid is associated.
`inheritMask`	Mask for an inherited overlay.
`extendMask`	Mask for an extended (appended) overlay.

5.15 Server object properties and structures

A *server object* is an object in the AR System server that can be discretely exported and whose change history can be tracked. AR System has the following server objects:

- Active links
- Character menus
- Containers

- Distributed mappings
- Escalations
- Fields
- Filters
- Forms

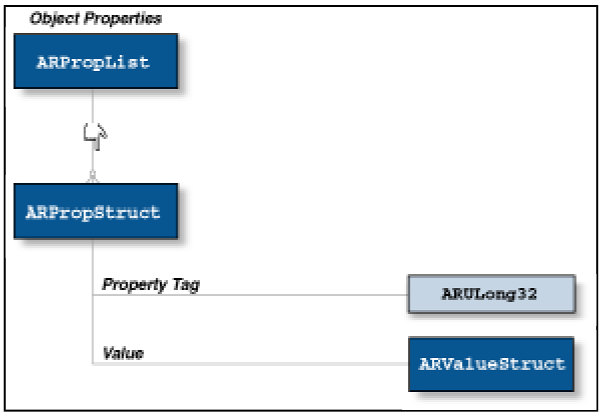
The `ARCreate Object`, `ARGet Object`, and `ARSet Object` functions for server objects have an `objPropList` parameter that defines the set of one or more custom properties (`AR_OPROP_*`) associated with an object.

To define server object properties, use tag-value pairs (see Server object property tags). The C API uses the `ARPropStruct` data structure to store the tag and an `ARValueStruct`, which stores the tag value. `ARPropStruct` structures are stored in the `ARPropList` data structure (see the following figure).

Note

`ARPropStruct` structures also store display properties (`AR_DPROP_*`) for objects such as fields. See Defining field display properties.

ARPropList data structure for server object properties



If the `objPropList` parameter is set to `NULL`, an object property list with zero properties is associated with the server object.

5.15.1 Server object property tags

You define server object property tags in the following file: ***ARSystemInstallDir*\ Arserver\api\include\ar.h** file.

Tags with a number of 60000 (`AR_OPROP_RESERVED`) or less are reserved. API programmers can define any server object property tag with a number greater than `AR_OPROP_RESERVED`, but the number of the last reserved tag might increase in a later release, requiring a program update.

The following server object property tags are predefined in the AR System server. You can specify them in any order in a server object property list (`ARPropList`). You can also include multiple instances of the same tag in a property list; however, API programs might ignore duplicate instances of properties.

`AR_OPROP_VENDOR_NAME: (CHAR):`
Name of the company that produces the product, for example, "Arrow Systems, Inc."

`AR_OPROP_VENDOR_PRODUCT (CHAR):`
Product name, for example, "BMC Remedy AR System."

`AR_OPROP_VENDOR_VERSION: (CHAR):`
Product version, for example, "7.5.00."

`AR_OPROP_GUID: (CHAR):`
Globally unique identifier for the object.

`AR_OPROP_COPYRIGHT: (CHAR):`
Copyright string.

`AR_OPROP_POOL_NUMBER: (INT):`
Pool number for an escalation if the object is assigned to a pool.

`AR_OPROP_CORE_FIELDS_OPTION_MASK: (INT):`
Bitmask of options for core fields of the schema. A value of `AR_CORE_FIELDS_OPTION_NONE` means no options are set. A value of `AR_CORE_FIELDS_OPTION_DISABLE_STATUS_HISTORY` means that status history recording and retrieval is disabled for the schema.

`AR_OPROP_CACHE_DISP_PROP: (INT):`
Bitmask of options that control whether VUI and field display properties are cached for the schema. If this property is absent, the server uses the default specified by the `AR_SERVER_INFO_CACHE_DISP_PROP` property.

- No bits — Do not cache VUI or field display properties (`AR_CACHE_DPROP_NONE`).
- Bit 1 — Cache VUI display properties (`AR_CACHE_DPROP_VUI`).
- Bit 2 — Cache field display properties (`AR_CACHE_DPROP_FIELD`).
- Bits 1 and 2 — Cache VUI and field display properties (`AR_CACHE_DPROP_ALL`).

`AR_OPROP_OVERLAY_EXTEND_MASK: (INT):`
Bitmask options that indicate what is extended in an overlay. The options are:

1	`AR_GRANULAR_PERMISSIONS`	Extends permissions. You can use this bit for forms, fields, active links, active link guides, applications, and packing lists.
2	`AR_GRANULAR_FORM_LIST`	Extends forms to which this workflow is attached. You can use this bit for active links, active link guides, filters, filter guides, and escalations.
4	`AR_GRANULAR_INDEX`	Extends indexes. You can use this bit for forms.

8	AR_GRANULAR_OTHER	No object currently allows extension of the other properties.

AR_OPROP_OVERLAY_INHERIT_MASK: (INT):
Bitmask options that indicate what is inherited in an overlay. The options are:

1	AR_OVERLAY_PERMISSIONS	Inherits permissions. You can use this bit for forms, fields, active links, active link guides, applications and packing lists.
2	AR_OVERLAY_FORM_LIST	Inherits forms to which this workflow is attached. You can use this bit for active links, active link guides, filters, filter guides, and escalations.
4	AR_OVERLAY_INDEX	Inherits indexes. You can use this bit for forms.
8	AR_OVERLAY_OTHER	Inherits everything that is not included in the previous bits.

Note

The AR_OPROP_OVERLAY_EXTEND_MASK and AR_OPROP_OVERLAY_INHERIT_MASK masks are mutually exclusive. Overwritten parts of the granular overlay are defined by which bits are not set in either mask. Anything that is not extended or inherited is overwritten.

Set and Get functions access the masks through object properties. You must inherit or extend indexes. They cannot be overwritten, so the AR_OVERLAY_INDEX bit must appear in the AR_OPROP_OVERLAY_EXTEND_MASK or AR_OPROP_OVERLAY_INHERIT_MASK object property.

5.15.2 Server-managed object property tags

The BMC Remedy AR System server has the following object property tags (AR_SMOPROP_*) that are managed exclusively by the server.

AR_SMOPROP_APP_OWNER
Name of the application owner for the object.

AR_SMOPROP_OBJECT_LOCK_TYPE
Object lock type.The following table lists valid values for the lock type.

0	AR_LOCK_TYPE_NONE	Export the objects as unlocked.
1	AR_LOCK_TYPE_READONLY	Export the objects with read-only lock. When the objects are imported, they are readable but not modifiable.
2	AR_LOCK_TYPE_HIDDEN	Export the objects with hidden lock. When the objects are imported, they are neither readable nor modifiable.

AR_SMOPROP_OBJECT_LOCK_KEY
Object lock key. This string is used as a key (or password) to enforce locking.

`AR_SMOPROP_ENTRYPOINT_DEFAULT_NEW_ORDER`
Entry point order in entry point list.

`AR_SMOPROP_ENTRYPOINT_DEFAULT_SEARCH_ORDER`
Entry point order in entry point list.

`AR_SMOPROP_APP_STAT_FORM`
Integer that indicates whether the object participates in application statistics.

`AR_SMOPROP_APP_LIC_VERSION`
License version for applications and forms.

`AR_SMOPROP_APP_LIC_DESCRIPTOR`
License descriptor for applications and forms.

`AR_SMOPROP_APP_LIC_USER_LICENSABLE`
Numeric Boolean value that indicates whether the form is user licensable.

`AR_SMOPROP_APP_ACCESS_POINT`
Integer that indicates whether the object is an access point. Zero (0) denotes that the object is not an access point. One (1) denotes that the object is an access point.

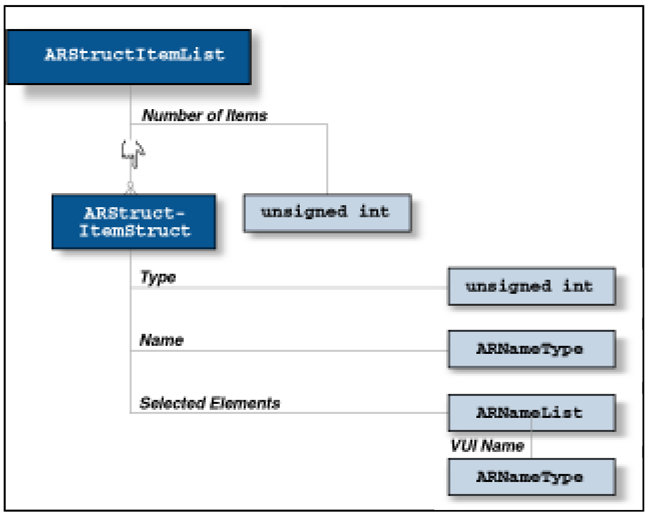
`AR_SMOPROP_APP_BSM_TAG`
For internal use only.

`AR_SMOPROP_PRIMARY_FIELDSET`
Set of fields used as a unique index for importing application-specific data. This property applies only if the schema includes fields 61000 and 61001.

5.16 Importing and exporting and structures

Both the ARExport and ARImport functions use the `ARStructItemList` structure to represent BMC Remedy AR System objects to export or import. This list contains zero or more `ARStructItemStruct` items (see the following figure).

Structures used to import and export BMC Remedy AR System objects



Each `ARStructItemStruct` represents a particular object to export or import:

`Item Type`	Integer that specifies the type of BMC Remedy AR System object (see the following table).
`Item Name`	Name associated with the BMC Remedy AR System object.
`Selected Elements`	During export and import operations, this parameter is used only when processing data of the following types: • `AR_STRUCT_ITEM_SCHEMA_MAIL` — VUI mail template to export. • `AR_STRUCT_ITEM_VUI` — VUIs to export from or import into the schema. Specify the schema name in the name parameter. If the selected elements list is empty (zero elements), all schema VUIs are exported or imported. • `AR_STRUCT_ITEM_LOCALE_VUI` — Locales of the VUIs to export with the schema. Specify the schema name in the name parameter. If the selected elements list is empty (zero elements), schema VUIs for all locales are exported.
`VUI Name`	Schema view name. An empty string specifies views with no defined locales.

5.16.1 Item type values for ARStructItemStruct

BMC Remedy AR System format

1	AR_STRUCT_ITEM_SCHEMA	Schema definition, including views, help text, and change diary information.
2	AR_STRUCT_ITEM_SCHEMA_DEFN *****	Structure definition for configuring BMC Remedy User cache (special purpose export).
3	AR_STRUCT_ITEM_SCHEMA_VIEW *****	

		Display definition for configuring BMC Remedy User cache (special purpose export).
4	AR_STRUCT_ITEM_SCHEMA_MAIL	Email template for submitting requests (special purpose export).
5	AR_STRUCT_ITEM_FILTER	Filter definition.
6	AR_STRUCT_ITEM_ACTIVE_LINK	Active link definition.
7	AR_STRUCT_ITEM_ADMIN_EXT	No longer supported.
8	AR_STRUCT_ITEM_CHAR_MENU	Character menu definition.
9	AR_STRUCT_ITEM_ESCALATION	Escalation definition.
10	AR_STRUCT_ITEM_DIST_MAP	Mapping definition for Distributed Server Option (DSO).
11	AR_STRUCT_ITEM_SCHEMA_VIEW_MIN *****	Display definition without bitmaps for BMC Remedy User.
12	AR_STRUCT_ITEM_CONTAINER	Container definition.
13	AR_STRUCT_ITEM_DIST_POOL	Pool definition for the attachments.
14	AR_STRUCT_ITEM_VUI	View definition.
15	AR_STRUCT_ITEM_FIELD	Field definition. This item is supported only in the XML API.
16	AR_STRUCT_ITEM_APP	Application definition.
18	AR_STRUCT_ITEM_LOCALE_VUI	Local-based views.
30	AR_STRUCT_ITEM_SCHEMA_DATA	Schema data definition.
103	AR_STRUCT_ITEM_SCHEMA_VIEW_2	Definition tag for all schema views instead of just one.

Values marked with an asterisk (*****) are available for legacy environments that use BMC Remedy User, which is no longer shipped with BMC Remedy AR System.

XML format

1073741825	AR_STRUCT_ITEM_XML_SCHEMA	Schema definition, including views, help text, and change diary information.
1073741829	AR_STRUCT_ITEM_XML_FILTER	Filter definition.
1073741830	AR_STRUCT_ITEM_XML_ACTIVE_LINK	Active link definition.
1073741831	AR_STRUCT_ITEM_XML_ADMIN_EXT	No longer supported.
1073741832	AR_STRUCT_ITEM_XML_CHAR_MENU	Character menu definition.
1073741833	AR_STRUCT_ITEM_XML_ESCALATION	Escalation definition.

1073741834	AR_STRUCT_ITEM_XML_DIST_MAP	Mapping definition for DSO.
1073741836	AR_STRUCT_ITEM_CONTAINER	Container definition.
1073741837	AR_STRUCT_ITEM_DIST_POOL	Pool definition for the attachments.
1073741838	AR_STRUCT_ITEM_XML_VUI	View definition.
1073741839	AR_STRUCT_ITEM_XML_FIELD	Field definition. This item is supported only in the XML API.
1073741840	AR_STRUCT_ITEM_XML_APP	Application definition.
1073741842	AR_STRUCT_ITEM_XML_LOCALE_VUI	Local-based views.

To export a form that you want to import to another server, you must specify `AR_STRUCT_ITEM_SCHEMA`. The three partial form types do not contain complete form definitions and are used only for caching purposes.

`The AR_STRUCT_XML_OFFSET` (1 << 30) bitmask is used to derive the XML structure item types. The XML type is the logical-OR of the non-XML type and the XML offset. For example, `AR_STRUCT_ITEM_XML_SCHEMA` is defined in the **ar.h** file as (`AR_STRUCT_XML_OFFSET |` `AR_STRUCT_ITEM_SCHEMA`).

Note

To import all structures in the import buffer, specify `NULL` for the `structItems` parameter or an `ARStructItemList` with zero items.

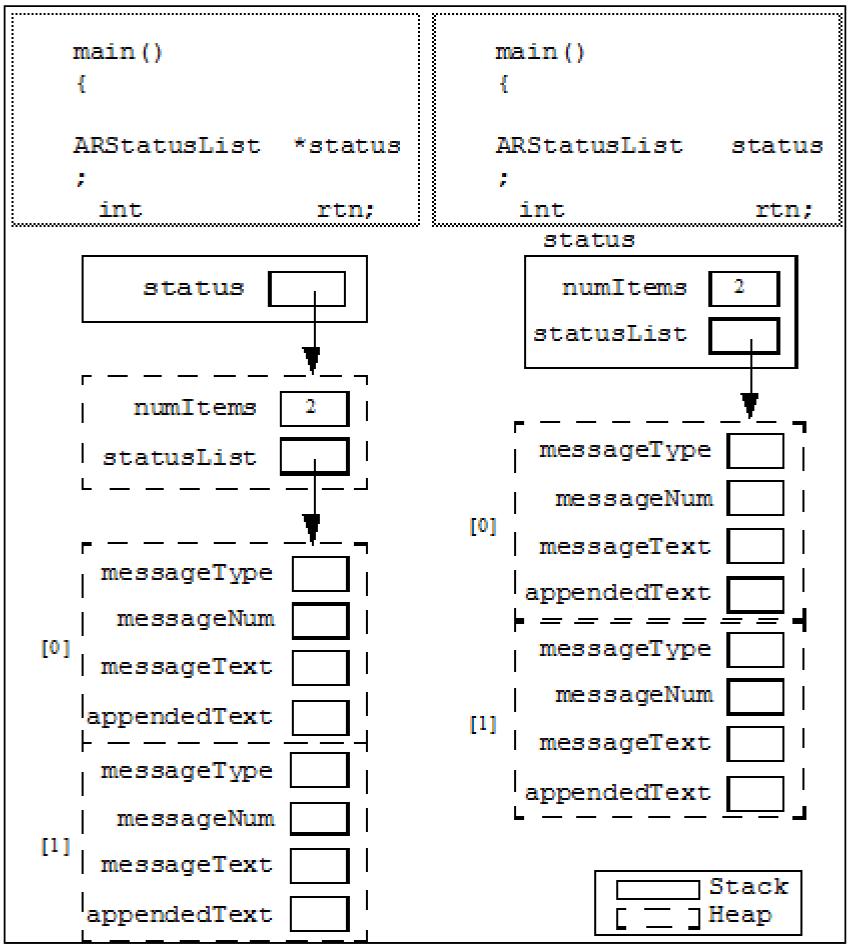
5.17 Freeing allocated memory

Many of the BMC Remedy AR System data structures use allocated memory. For *input parameters*, use the `malloc` function to allocate the necessary space. The system dynamically allocates memory for *output parameters* based on their content. Only you (the caller of the API function) know whether you allocated space for input parameters, and when you are finished with output parameters, you are responsible for freeing *all* allocated memory. To avoid a buildup of allocated space, free memory as soon as you no longer need it.

To ease the process of freeing memory, the API includes `FreeAR` functions, each specialized to free all allocated memory associated with a particular data structure. These functions assume that all structure components are in allocated memory and must not be used if any components of a structure are on the stack. In that case, you must use the `free` function to free the memory manually.

All `FreeAR` functions accept these input arguments:

- `value` — Pointer to the structure to free. The function recursively frees all allocated memory in the structure. If you set this to `NULL` (or the structure is a list with zero items), the function performs no operations.
- `freeStruct` — Boolean value that specifies whether to free the top-level structure. If you allocated memory for the top-level structure, specify `TRUE` to free both the structure and its contents. If you used a stack variable for the top-level structure, specify `FALSE` to free only the contents of the structure. The following example illustrates this difference.

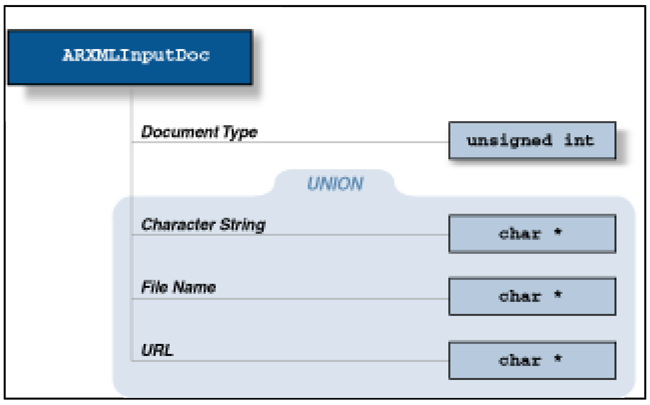


When an API function fails (that is, `status` is greater than one), the system initializes all output parameters except `ARStatusList`. As a result, the only `FreeAR` function the program *must* call is `FreeARStatusList`. Calling `FreeAR` *data Structure* for the other parameters is unnecessary but harmless.

5.18 XML formats

To transform XML documents, use the `ARXMLInputDoc` and `ARXMLOutputDoc` formats.

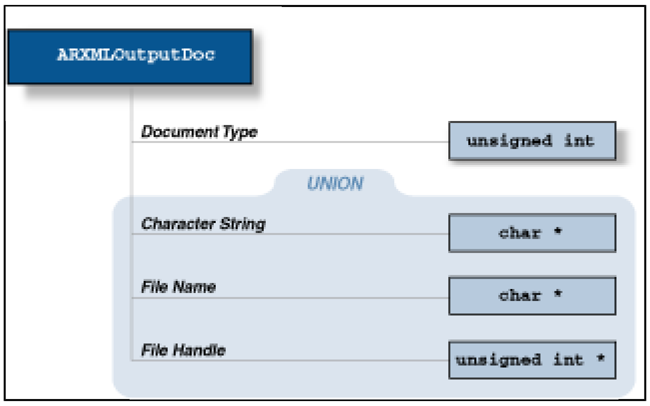
ARXMLInputDoc format



The ARXMLInputDoc format, which transforms XML documents to BMC Remedy AR System structures, has the following elements:

Document Type	Type of XML documents.
Character String	Null-terminated character string of the XML document (AR_XML_CHAR_STR).
File Name	File name of the XML document (AR_XML_DOC_FILE_NAME).
URL	URL of the XML document (AR_XML_DOC_URL).

ARXMLOutputDoc format



The ARXMLOutputDoc format, which transforms BMC Remedy AR System structures to XML documents, has the following elements:

Document Type	Type of XML documents.
Character String	Null-terminated character string of the XML document ( ).
File Name	File name of the XML document ( ).
File Handle	File handle of the XML document ( ).

6 BMC Remedy AR System C API functions

This section describes the BMC Remedy AR System C API functions.

The BMC Remedy AR System server uses API functions to perform operations on all BMC Remedy AR System objects. The **arextern.h** file contains the API function definitions. The **arfree.h** file contains the definitions of associated routines that free allocated memory see Freeing allocated memory.

The following topics are provided:

- Types of functions
- Function descriptions

6.1 Types of functions

The API functions consist of object manipulation functions, notification functions, and general use functions, such as functions that initialize or terminate sessions, authentication functions, and functions that import or export object definitions.

Use the following information to know more about:

- Object manipulation functions
- Alert functions
- Other functions

6.1.1 Object manipulation functions

You can perform five primary operations (*create*, *delete*, *get* (retrieve), *get list* (retrieve a list), and *set* (modify)) for each of the following objects:

- Active links
- Containers
- Entries
- Escalations
- Fields
- Filters
- Character menus
- Forms (schemas)
- Support files
- VUIs
- Images

For example, you can retrieve field properties or create entries for a specified form. Users with administrator capability can create or delete forms or other objects on a specified server. Some server objects can also be manipulated in other ways by additional functions. Specifically, these are the BMC Remedy AR System functions associated with each object.

Note

Functions that are used to perform administrative operations are noted by an asterisk (*).

Active links

- `ARCreateActiveLink` *
- `ARDeleteActiveLink` *
- `ARGetActiveLink` *
- `ARGetListActiveLink` *
- `ARGetMultipleActiveLinks` *
- `ARSetActiveLink` *

Containers

- `ARCreateContainer` *
- `ARDeleteContainer` *
- `ARGetContainer`
- `ARGetListContainer`
- `ARGetMultipleContainers`
- `ARSetContainer` *

Entries

- `ARCreateEntry`
- `ARDeleteEntry` *
- `ARGetEntry`
- `ARGetEntryBLOB`
- `ARGetEntryBlock`
- `ARGetEntryStatistics`
- `ARGetListEntry`
- `ARGetListEntryBlocks`
- `ARGetListEntryWithFields`
- `ARGetListEntryWithMultiSchemaFields`
- `ARGetMultipleEntries`
- `ARMergeEntry`
- `ARSetEntry`

Escalations

- `ARCreateEscalation` *
- `ARDeleteEscalation` *
- `ARGetEscalation`
- `ARGetListEscalation`
- `ARGetMultipleEscalations`
- `ARSetEscalation` *

Fields

- `ARCreateField` *
- `ARCreateMultipleFields` *
- `ARDeleteField` *
- `ARDeleteMultipleFields` *
- `ARGetField`
- `ARGetListEntryWithFields`
- `ARGetListField`
- `ARGetMultipleExtFieldCandidates`
- `ARGetMultipleFields`
- `ARSetField` *
- `ARSetMultipleFields` *

Filters

- `ARCreateFilter` *
- `ARDeleteFilter` *
- `ARGetFilter`
- `ARGetListFilter`
- `ARGetMultipleFilters`
- `ARSetFilter` *

Character menus

- `ARCreateCharMenu` *
- `ARDeleteCharMenu` *
- `ARExpandCharMenu`
- `ARGetCharMenu`
- `ARGetListCharMenu`
- `ARGetMultipleCharMenus`
- `ARSetCharMenu` *

Forms (schemas)

- `ARCreateSchema` *

- `ARDeleteSchema` *
- `ARGetListExtSchemaCandidates`
- `ARGetListSchema`
- `ARGetListSchemaWithAlias`
- `ARGetMultipleSchemas`
- `ARGetSchema`
- `ARSetSchema` *

Support files

- `ARCreateSupportFile` *
- `ARDeleteSupportFile` *
- `ARGetListSupportFile`
- `ARGetSupportFile`
- `ARSetSupportFile` *

VUIs

- `ARCreateVUI` *
- `ARDeleteVUI` *
- `ARGetListVUI`
- `ARGetMultipleVUIs`
- `ARGetVUI`
- `ARSetVUI` *

Images

- `ARCreateImage` *
- `ARGetImage`
- `ARGetMultipleImages`
- `ARSetImage` *
- `ARDeleteImage` *
- `ARGetListImage`

6.1.2 Alert functions

The BMC Remedy AR System includes the following alert functions that register users, cancel user registration, enter alert events into the system, and retrieve a list of alert users.

- `ARCreateAlertEvent`
- `ARDecodeAlertMessage`
- `ARDeregisterForAlerts`

- `ARGetAlertCount`
- `ARGetListAlertUser`
- `ARRegisterForAlerts`

6.1.3 Other functions

The API also includes functions for a variety of other operations, such as localization, data structure, housekeeping, object definition, server processes, and text manipulation.

Note

Functions that are used to perform administrative operations are noted by an asterisk (*).

Access control

- `ARGetListGroup`
- `ARGetListRole` *
- `ARGetListUser`
- `ARSetImpersonatedUser`
- `ARValidateFormCache`
- `ARVerifyUser`

Bulk entry

- `ARBeginBulkEntryTransaction`
- `AREndBulkEntryTransaction`

Currency

- `ARGetCurrencyRatio`
- `ARGetMultipleCurrencyRatioSets`

Data structure help functions

- `ARDateToJulianDate`
- `ARJulianDateToDate`
- `ARDecodeARAssignStruct`
- `ARDecodeARQualifierStruct`
- `ARDecodeDiary`
- `ARDecodeStatusHistory`
- `AREncodeARAssignStruct`
- `AREncodeARQualifierStruct`

- `AREncodeDiary`
- `AREncodeStatusHistory`
- `ARLoadARQualifierStruct`

Housekeeping

- `ARInitialization`
- `ARTermination`

Licensing

- `ARCreateLicense`
- `ARGetListLicense`
- `ARDeleteLicense` *
- `ARExportLicense` *
- `ARImportLicense` *
- `ARValidateLicense`
- `ARValidateMultipleLicenses`

Object definitions and external data manipulation

- `ARExport`
- `ARExportToFile`
- `ARImport` *
- `ARGetListSQL`

Server processes

- `ARGetListServer`
- `ARGetLocalizedValue`
- `ARGetMultipleLocalizedValues`
- `ARGetServerInfo`
- `ARGetServerStatistics`
- `ARExecuteProcess` *
- `ARSetLogging`
- `ARSetServerInfo` *
- `ARSetServerPort`

Text manipulation

- `ARGetTextForErrorMessages`

6.2 Function descriptions

This section describes each API function and its components.

Note

In API function descriptions, when a value is said to be specified, it means the value is found within the synopsis for that particular API function.

6.2.1 ARBeginBulkEntryTransaction

Description

Marks the beginning of a series of entry API function calls that are grouped together and sent to the BMC Remedy AR System server as part of one transaction. All calls related to create, set, delete, and merge operations made between the call to this function and a trailing call to `AREndBulkEntryTransaction` are not sent to the server until the trailing call is made.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARBeginBulkEntryTransaction(
ARControlStruct *control,
ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

AREndBulkEntryTransaction.

6.2.2 ARCreateActiveLink

Description

Creates a new active link with the indicated name on the specified server. The active link is added to the server immediately and returned to users who request information about active links. Because active links operate on clients, individual clients do not receive the new definition until they reconnect to the form (thus reloading the form from the server).

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateActiveLink(
ARControlStruct *control,
ARNameType name,
unsigned int order,
ARWorkflowConnectStruct *schemaList,
ARInternalIdList *groupList,
unsigned int executeMask,
ARInternalId *controlField,
ARInternalId *focusField,
unsigned int enable,
ARQualifierStruct *query,
ARActiveLinkActionList *actionList,
ARActiveLinkActionList *elseList,
char *helpText,
ARAccessNameType owner,
char *changeDiary,
ARPropList *objPropList,
unsigned int errorActlinkOptions,
ARNameType errorActlinkName,
char *objectModificationLogLabel,
char *taskName,
ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARCreate*` function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the active link to create. The names of all active links on a given server must be unique.

order

A value between `0` and `1000` (inclusive) that determines the active link execution order. When multiple active links are associated with a form, the value associated with each active link determines the order in which they are processed (from lowest to highest).

schemaList

The list of form names the active link is linked to. The active link must be associated with a single form or a list of forms that currently exists on the server.

groupList

A list of zero or more groups who can access this active link. Users can execute an active link if they belong to a group that has access to it. Specify an empty group list to define an active link accessible only by users with administrator capability. Specify group ID `0` (Public) to provide access to all users.

executeMask

A bitmask that indicates the form operations that trigger the active link.

Bit	Symbol	Description
0	AR_EXECUTE_ON_BUTTON	Execute the active link when a user selects a button, toolbar button, or menu item specified by the controlField parameter.
1	AR_EXECUTE_ON_RETURN	Execute the active link when a user presses Return in field specified by the focusField parameter.
2	AR_EXECUTE_ON_SUBMIT	Execute the active link when a user submits an entry (*before* data is sent to the BMC Remedy AR System server).
3	AR_EXECUTE_ON_MODIFY	Execute the active link when a user modifies an individual entry (*before* data is sent to the BMC Remedy AR System server).

4	AR_EXECUTE_ON_DISPLAY	Execute the active link when a user displays an entry (after data is retrieved from the BMC Remedy AR System server).
7	AR_EXECUTE_ON_MENU_CHOICE	Execute the active link when a user selects an item from a character menu associated with a field specified by the focusField parameter or selects a row in a table field specified by the focusField parameter.
9	AR_EXECUTE_ON_SET_DEFAULT	Execute the active link when a user sets default values (either manually or through preference settings).
10	AR_EXECUTE_ON_QUERY	Execute the active link when a user retrieves one or more entries (*before* the query is sent to the BMC Remedy AR System server).
11	AR_EXECUTE_ON_AFTER_MODIFY	Execute the active link when a user modifies an individual entry (*after* data is committed to the database).
12	AR_EXECUTE_ON_AFTER_SUBMIT	Execute the active link when a user submits an entry (*after* data is committed to the database).
14	AR_EXECUTE_ON_WINDOW_OPEN	Execute the active link when a user opens any form window or changes its mode.
15	AR_EXECUTE_ON_WINDOW_CLOSE	Execute the active link when a user closes any form window or changes its mode.

controlField

The ID of the field that represents the button, toolbar button, or menu item associated with executing the active link. This parameter is ignored if you do not specify the `AR_EXECUTE_ON_BUTTON` condition (see the `executeMask` parameter on executeMask).

focusField

The ID of the field associated with executing the active link by pressing Return, selecting a character menu item, or gaining or losing focus. You must create another active link to specify a different field for each condition. This parameter is ignored if you do not specify one of the following conditions: `AR_EXECUTE_ON_RETURN`, `AR_EXECUTE_ON_MENU_CHOICE`, `AR_EXECUTE_ON_LOSE_FOCUS`, or `AR_EXECUTE_ON_GAIN_FOCUS` (see the `executeMask` parameter on executeMask).

enable

A flag to enable or disable this active link. A value of `0` disables the active link, causing it to be invisible to the user and unavailable for use. A value of **1** enables the active link, causing it to be visible and available for use.

query

A qualification that determines whether the active link is executed. Specify `NULL` or assign an operation value of `0` (`AR_COND_OP_NONE`) to execute the active link unconditionally.

actionList

The set of actions performed if the condition defined by the query parameter is satisfied. You can specify from `1` to `25` actions in this list (limited by `AR_MAX_ACTIONS`).

elseList

The set of actions performed if the condition defined by the query parameter is not satisfied. You can specify from `0` to `25` actions in this list (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter (or zero actions) if you do not want to define any else actions.

helpText

The help text associated with the active link. This text can be of any length. Specify `NULL` for this parameter if you do not want to associate help text with this object.

owner

The owner for the active link. The owner defaults to the user performing the operation if you specify `NULL` for this parameter.

changeDiary

The initial change diary associated with the active link. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a properties list with zero properties is associated with the object, and zero properties are returned when an `ARGetActiveLink` is performed. See Server object properties.

errorActlinkOptions

Reserved for future use. Set to `NULL`.

errorActlinkName

Reserved for future use. Set to `NULL`.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDeleteActiveLink, ARDeleteField, ARGetActiveLink, ARGetField, ARGetListActiveLink, ARGetListField, ARGetMultipleActiveLinks, ARSetActiveLink, ARSetField. See FreeAR for: FreeARActiveLinkActionList, FreeARInternalIdList, FreeARQualifierStruct, FreeAR, FreeARPropList, FreeARWorkflowConnectStruct.

6.2.3 ARCreateAlertEvent

Description

Enters an alert event on the specified server. The BMC Remedy AR System server sends an alert to the specified, registered users.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateAlertEvent(
ARControlStruct *control,
ARAccessNameType user,
char *alertText,
int priority,
ARNameType sourceTag,
ARServerNameType serverName,
ARNameType formName,
char *objectId,
AREntryIdType *entryId,
ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

user

The user who receives the alert. Specify `*` (`AR_REGISTERED_BROADCAST`) to create an alert event for all users that are currently registered to receive alerts with the BMC Remedy AR System server. You cannot specify a group name for this argument.

alertText

The text that the alert contains.

priority

A relative value that represents the priority for this alert. The range of acceptable values is between 0 and 10.

sourceTag

A string that identifies the source of the alert. The BMC Remedy AR System provides two predefined values for this string:

- `AR` — alert originated from the BMC Remedy AR System
- `FB` — alert originated from Flashboards

serverName

The name of the server that is the source of the alert. Use `@` to specify the current server. Specify `NULL` for this parameter if the parameter is not applicable to the type of alert that this call creates.

formName

The name of the form that is the source of the alert. For Flashboards, this is the name of the Flashboard that generated the alert. Specify `NULL` for this parameter if the parameter is not applicable to the type of alert that this call creates.

objectId

The identifier for the object. For BMC Remedy AR System, this value is the entry ID of the originating request. For Flashboards, this value is the name of the Flashboard alert that the user provides. Specify `NULL` for this parameter if the parameter is not applicable to the type of alert that this call creates. Join forms have multiple entry Ids (**000000000000001|000000000000002**) separated by vertical bars.

Return values

entryId

The unique identifier for the new alert (system-generated).

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARRegisterForAlerts, ARDeregisterForAlerts. See FreeAR for: FreeARStatusList.

6.2.4 ARCreateCharMenu

Description

Creates a new character menu with the indicated name on the specified server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateCharMenu(
ARControlStruct *control,
ARNameType name,
unsigned int refreshCode,
ARCharMenuStruct *menuDefn,
char *helpText,
ARAccessNameType owner,
char *changeDiary,
ARPropList *objPropList,
char b*objectModificationLogLabel,
char *taskName,
ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARCreate*` function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the character menu to create. The names of all character menus on a given server must be unique.

refreshCode

A value that indicates when the menu is refreshed. This parameter enables you to balance menu consistency with performance.

1	AR_MENU_REFRESH_CONNECT	Refresh only when form opened.
2	AR_MENU_REFRESH_OPEN	Refresh every time menu opened.
3	AR_MENU_REFRESH_INTERVAL	Refresh first time menu opened and every 15 minutes thereafter.

menuDefn

The definition of the character menu.

helpText

The help text associated with the character menu. This text can be of any length. Specify `NULL` for this parameter if you do not want to associate help text with this object.

owner

The character menu owner. The owner defaults to the user performing the operation if you specify `NULL` for this parameter.

changeDiary

The initial change diary associated with the character menu. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a properties list with zero properties is associated with the character menu, and a list of zero properties is returned when an `ARGetCharMenu` is performed. See Server object properties.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDeleteCharMenu, ARExpandCharMenu, ARGetCharMenu, ARGetListCharMenu, ARSetCharMenu. See FreeAR for: FreeARCharMenuStruct, FreeARStatusList, FreeARPropList.

6.2.5 ARCreateContainer

Description

Creates a new container with the indicated name on the specified server. Use this function to create applications, active links, active link guides, filter guide, packing lists, guides, and BMC Remedy AR System-defined container types. A container can also be a custom type that you define.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateContainer(
ARControlStruct *control,
ARNameType name,
ARPermissionList *groupList,
ARInternalIdList *admingrpList,
ARContainerOwnerObjList *ownerObjList,
char *label,
char *description,
unsigned int *type,
ARReferenceList *references,
ARBoolean removeFlag,
char *helpText,
ARAccessNameType owner,
char *changeDiary,
ARPropList *objPropList,
char *objectModificationLogLabel,
char *taskName,
ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARCreate*` function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the container to create. The names of all containers on a given server must be unique.

groupList

A list of zero or more groups who can access this container. Specify an empty group list to define a container accessible only by users with administrator capability. Specify group ID **0** (Public) to provide access to all users. The permission value that you assign for each group determines whether users in that group see the container in the container list.

1	AR_PERMISSIONS_VISIBLE	Users see the container in the container list.
2	AR_PERMISSIONS_HIDDEN	Users do not see the container in the container list.

admingrpList

A list of zero or more groups who can administer this container (and the referenced objects). If `ownerObj` is not `NULL`, this parameter is ignored and the Subadministrator group list of the owning form is used instead. Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify an empty administrator group list to define a container that can only be administered by users with administrator capability. Specify group ID `0` (Public) to provide administrator capability to all members of the Subadministrator group.

ownerObjList

A list of schemas that own this container. This parameter can be `NULL` if the container exists globally.

label

The label for this container. It can be as many as 255 characters long or `NULL`.

description

The description for this container. It can be as many as 2000 characters long or `NULL`.

type

The type for this container — either guide (`ARCON_GUIDE`), application (`ARCON_APP`), or a custom type that you have defined.

references

A list of pointers to the objects referenced by this container. References can be to internal BMC Remedy AR System objects (for example, guides reference active links and applications reference forms) or to external objects such as URLs or file names. Specify `NULL` for this parameter if you do not want to associate any objects with this container.

removeFlag

A flag that specifies how invalid object references are removed when the container is created. If `FALSE`, references to nonexistent BMC Remedy AR System objects are removed with no error generated. If `TRUE`, an error is generated.

helpText

The help text associated with the container. This text can be of any length. Specify `NULL` for this parameter if you do not want to associate help text with this object.

owner

The owner for the container. The owner defaults to the user performing the operation if you specify `NULL` for this parameter.

changeDiary

The initial change diary associated with the container. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a properties list with zero properties is associated with the container, and a list of zero properties is returned when an `ARGetContainer` is performed. See Server object properties.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateSchema, ARDeleteContainer, ARGetContainer, ARGetListContainer, ARGetListEntry, ARSetContainer. See FreeAR for: FreeARContainerInfoList, FreeARInternalIdList, FreeARPermissionList, FreeARPropList, FreeARReferenceList, FreeARStatusList.

6.2.6 ARCreateEntry

Description

Creates a new entry in the indicated schema (form) on the specified server. You can only create entries in base schemas. To add entries to join forms, create them in one of the underlying base forms.

Privileges

The system creates data based on the access privileges of the user that you specify for the `control` parameter and the **createMode** setting for each field (see ARCreateField). User permissions are verified for each specified field. The system generates an error if the user does not have write permission for a field or a field does not exist.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateEntry(
ARControlStruct *control,
ARNameType schema,
ARFieldValueList *fieldList,
AREntryIdType *entryId,
ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the schema to create the entry in.

fieldList

A list of one or more field/value pairs (specified in any order) that identifies the data for the new entry. You must specify values for all required fields that do not have defined defaults. Values must be of the data type defined for the field or have a data type of `0` (`AR_DATA_TYPE_NULL`). `NULL` values can only be specified for optional fields, and assigning `NULL` overrides any defined field default. An error is generated if a field does not exist or the user specified by the `control` parameter does not have write permission for a field.

Return values

entryId

The unique identifier for the new entry (system-generated).

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDeleteEntry, ARGetEntry, ARGetListEntry, ARMergeEntry, ARSetEntry. See FreeAR for: FreeARFieldValueList, FreeARStatusList.

6.2.7 ARCreateEscalation

Description

Creates a new escalation with the indicated name on the specified server. The escalation condition is checked regularly based on the time structure defined when it is enabled.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateEscalation(
ARControlStruct *control,
ARNameType name,
AREscalationTmStruct *escalationTm,
ARWorkflowConnectStruct *schemaList,
unsigned int enable,
ARQualifierStruct *query,
ARFilterActionList *actionList,
ARFilterActionList *elseList,
char *helpText,
ARAccessNameType owner,
char *changeDiary,
ARPropList *objPropList,
char *objectModificationLogLabel,
char *taskName,
ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARCreate*` function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the escalation to create. The names of all escalations on a given server must be unique.

escalationTm

The time specification for evaluating the escalation condition. This parameter can take one of two forms: a time interval that defines how frequently the server checks the escalation condition (in seconds) or a bitmask that defines a particular day (by month or week) and time (hour and minute) for the server to check the condition.

schemaList

The list of form names the escalation is linked to. The escalation must be associated with a single form or a list of forms that currently exists on the server.

enable

A flag to enable or disable this escalation. A value of `0` disables the escalation, causing its condition checks and associated actions to not be performed. A value of `1` enables the escalation, causing its conditions to be checked at the specified time interval.

query

A query operation performed when the escalation is executed that determines the set of entries to which the escalation actions (defined by the `actionList` parameter) are applied. Specify `NULL` or assign an operation value of `0` (`AR_COND_OP_NONE`) to match all schema entries.

actionList

The set of actions performed for each entry that matches the criteria defined by the `query` parameter. You can specify from 1 to 25 actions in this list (limited by `AR_MAX_ACTIONS`).

elseList

The set of actions performed if no entries match the criteria defined by the `query` parameter. These actions are *not* performed for all non-matching entries. You can specify from 0 to 25 actions in this list (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter (or zero actions) if you do not want to define any **else** actions.

helpText

The help text associated with the escalation. This text can be of any length. Specify `NULL` for this parameter if you do not want to associate help text with this object.

owner

The owner for the escalation. The owner defaults to the user performing the operation if you specify `NULL` for this parameter.

changeDiary

The initial change diary associated with the escalation. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a properties list with zero properties is associated with the escalation, and a list of zero properties is returned when an `ARGetEscalation` is performed. See Server object properties.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDeleteEscalation, ARGetEscalation, ARGetListEscalation, ARSetEscalation. See FreeAR for: FreeARFilterActionList, FreeARQualifierStruct, FreeARStatusList, FreeARWorkflowConnectStruct, FreeARPropList.

6.2.8 ARCreateField

Description

Creates a new field with the indicated name on the specified server. Forms can contain data and nondata fields. Nondata fields serve several purposes. Trim fields enhance the appearance and usability of the form (for example, lines, boxes, or static text). Control fields provide mechanisms for executing active links (for example, menus, buttons, or toolbar buttons). Other nondata fields organize data for viewing (for example, panels and panel holders) or show data from another form (for example, tables and columns).

An active link can be associated with only one control field (you can, however, select any combination of screen appearances for that field). A particular control field, however, can be associated with multiple active links. While information about nondata fields is stored on the server, they do not require storage space in the underlying database.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateField(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalId *fieldId,
   ARBoolean reservedIdOK,
   ARNameType fieldName,
   ARFieldMappingStruct *fieldMap,
   unsigned int dataType,
   unsigned int option,
   unsigned int createMode,
   unsigned int fieldOption
   ARValueStruct *defaultVal,
   ARPermissionList *permissions,
   ARFieldLimitStruct *limit,
   ARDisplayInstanceList *dInstanceList,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The `user` and `server` fields are required.

If a valid overlay group is specified in the control record, the `ARCreate*` function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

schema

The name of the form the field is linked to. The field must be associated with a single form that currently exists on the server.

fieldId

The internal ID of the field to create. The IDs of all fields and form views (VUIs) associated with a given form must be unique. Specify `0` for this parameter if you want the system to generate and return the ID. Otherwise, specify a value between **53686891** 2 and **2147483647** (limited by `AR_MAX_RESERVED_FIELD_ID` in **arstruct.h**). If you specify a reserved ID, the system generates an error unless the `reservedIdOK` parameter is set to `1` (`TRUE`).

reservedIdOK

A flag that indicates whether you can create the field with a reserved ID. Specify `1` (`TRUE`) for this parameter to allow reserved IDs. Specify `0` (`FALSE`) if you want the system to generate an error when you specify a reserved ID for the `fieldId` parameter.

fieldName

The name of the field to create. The names of all fields and VUIs associated with a given form must be unique. The system uses this name when it creates the SQL view of the form for report writing purposes. The field name is used to identify a field in a workflow object more easily. The field name is different from the field label in that it is specific to a form, but not specific to a view of the form. See the `dInstanceList` parameter on dInstanceList for information about how to specify a field label.

fieldMap

A mapping to the underlying form in which to create the field. To create a field in a base form, specify a field type of `1` (`AR_FIELD_REGULAR`) in the structure. Otherwise, specify a field type of `2` (`AR_FIELD_JOIN`) and identify the member form (primary or secondary) and field ID for the new field. If the member form is also a join form, create field mappings in all nested join forms until you can map the field to an underlying base form. See Mapping fields in schemas.

dataType

The data type of the field to create.

2	AR_DATA_TYPE_INTEGER	Integer
3	AR_DATA_TYPE_REAL	Real
4	AR_DATA_TYPE_CHAR	Character
5	AR_DATA_TYPE_DIARY	Diary
6	AR_DATA_TYPE_ENUM	Selection
7	AR_DATA_TYPE_TIME	Date/time
10	AR_DATA_TYPE_DECIMAL	Fixed-point decimal. Values must be specified in C locale, for example 1234.56
11	AR_DATA_TYPE_ATTACH	Attachment
12	AR_DATA_TYPE_CURRENCY	Currency
31	AR_DATA_TYPE_TRIM	Trim
32	AR_DATA_TYPE_CONTROL	Control
33	AR_DATA_TYPE_TABLE	Table
34	AR_DATA_TYPE_COLUMN	Column
35	AR_DATA_TYPE_PAGE	Panel
36	AR_DATA_TYPE_PAGE_HOLDER	Panel holder

option

A flag that indicates whether users must enter a value in the field.

1:	Required (data fields only) (AR_FIELD_OPTION_REQUIRED).
2:	Optional (data fields only) (AR_FIELD_OPTION_OPTIONAL).
4:	Display-only (data fields only). Works like an optional field but doesn't write to the database (AR_FIELD_OPTION_DISPLAY).

createMode

A flag that indicates the permission status for the field when users submit entries. This parameter is ignored for display-only fields.

1:	

	Any user (including guest users) can enter data in the field when submitting (AR_FIELD_OPEN_AT_CREATE).
2:	Only users who have been granted permission can enter data in the field when submitting (AR_FIELD_PROTECTED_AT_CREATE).

fieldOption

A bitmask that indicates whether the field is to be audited or copied when other fields are audited.

Bit 0:	Audit this field. (AR_FIELD_BITOPTION_AUDIT)
Bit 1:	Copy this field when other fields in the form are audited. (AR_FIELD_BITOPTION_COPY)
Bit 2:	Indicates this field is for Log Key 1. (AR_FIELD_BITOPTION_LOG_KEY1)
Bit 3:	Indicates this field is for Log Key 2. (AR_FIELD_BITOPTION_LOG_KEY2)
Bits 2 and 3:	Indicates this field is for Log Key 3. (AR_FIELD_BITOPTION_LOG_KEY3)

defaultVal

The value to apply if a user submits an entry with no field value (only applicable to data fields). The default value can be as many as 255 bytes in length (limited by `AR_MAX_DEFAULT_SIZE`) and must be of the same data type as the field. Specify `NULL` (or `AR_DEFAULT_VALUE_NONE`) for trim, control, panel, and panel holder fields or if you do not want to define a default value.

permissions

A list of zero or more groups who can access this field. Specify an empty permission list to define a field accessible only by users with administrator capability. Specify group **ID 0** (Public) to provide access to all users. The permission value that you assign for each group determines whether users in that group can modify the field. See Field permission values for.

limit

The value limits for the field and other properties specific to the field's type. See Defining field limits for a description of the contained structure that applies to the type of field that you are creating. The limits and properties that you define must be of the same data type as the field. Specify `NULL` (or `AR_FIELD_LIMIT_NONE`) for trim and control fields or if you do not want to define any limits or properties.

dInstanceList

A list of zero or more display properties to associate with the field. You can define both display properties common to all form views (VUIs) and display properties specific to particular form views. The system includes the field in each view that you specify, regardless of whether you define any display properties for those views. If you do not specify a property for a particular view, the system uses the default value (if defined). Specify `NULL` for this parameter if you do not want to define any display properties.

helpText

The help text associated with the field. This text can be of any length. Specify `NULL` for this parameter if you do not want to associate help text with this object.

owner

The owner for the field. The owner defaults to the user performing the operation if you specify `NULL` for this parameter.

changeDiary

The initial change diary associated with the field. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a list with zero properties is associated with the object, and zero properties are returned when an `ARGetField` is performed. See Server object properties and structures.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

Display properties

You can specify the following display properties in any order. Avoid specifying a property more than once, because the system then selects one of your definitions at random.

Note

When using the **driver** program to call the functions in your API program, you need to specify the property number instead of the property name. The property number is listed for each display property name in the **ar.h** file in the **api/include** directory. See Using the driver program for more information.

- `AR_DPROP_ALIGN` (unsigned long): A value that indicates the vertical alignment of text within its bounding box.

1:	Top-aligned (AR_DVAL_ALIGN_TOP).
2:	Centered (AR_DVAL_ALIGN_MIDDLE) (default setting).
4:	Bottom-aligned (AR_DVAL_ALIGN_BOTTOM).

- `AR_DPROP_ATTACH_COLGRAD_COLOR` (character string): When `AR_DPROP_ATTACH_COLGRAD_TYPE` is set to a gradient type, this value specifies the main color in the effect.

- AR_DPROP_ATTACH_COLGRADBKG_COLOR (character string): When AR_DPROP_ATTACH_COLGRAD_TYPE is set to a gradient type, this value specifies the second color in the effect.
- AR_DPROP_ATTACH_COLGRAD_TYPE (unsigned long): A value that enables a color fill gradient effect for a background color of the attachment pool column header.
- AR_DPROP_ATTACH_COLHDRTXT_COLOR (character string): The color of the text of the attachment pool column header.
- AR_DPROP_ATTACH_FTRGRAD_COLOR (character string): When AR_DPROP_ATTACH_FTRGRAD_TYPE is set to a gradient type, this value specifies the main color in the effect.
- AR_DPROP_ATTACH_FTRGRADBKG_COLOR (character string): When AR_DPROP_ATTACH_FTRGRAD_TYPE is set to a gradient type, this value specifies the second color in the effect.
- AR_DPROP_ATTACH_FTRGRAD_TYPE (unsigned long): A value that enables a color fill gradient effect for a background color of the attachment pool footer.
- AR_DPROP_AUTO_REFRESH (unsigned long): A value that indicates whether automatic refresh is enabled (only applicable to table fields). Specify 0 to disable (AR_DVAL_AUTO_REFRESH_NONE) and 1 to enable (AR_DVAL_AUTO_REFRESH_TABLE_MAX). When automatic refresh is enabled, the contents of the table field are updated when the entry is displayed in a Modify window. When automatic refresh is disabled, the table appears empty until the user clicks within its borders to refresh it.
- AR_DPROP_BACKGROUND_MODE (unsigned long): A value that indicates whether the background of a field is opaque or transparent (only applicable to trim fields). Specify 0 for opaque (AR_DVAL_BKG_MODE_OPAQUE) or 1 for transparent (AR_DVAL_BKG_MODE_TRANSPARENT). The default value is 0 (opaque).
- AR_DPROP_BBOX (ARCoordList): A set of coordinates (X,Y) that specify the screen boundaries (bounding box) for the field (applicable to all fields except panel and column fields). The first coordinate pair identifies the upper-left corner for the field. The second coordinate pair identifies the lower-right corner (see AR_DPROP_COORDS).
- AR_DPROP_BUTTON_2D (unsigned long): A flag that indicates whether the button appears as a two-dimensional image with no border (only applicable to control fields). Valid values for this property are 1 (TRUE) and 0 (FALSE). The default value is 0 (three-dimensional with borders).
- AR_DPROP_BUTTON_IMAGE_POSITION (unsigned long): A value that determines the position of the button image relative to the button label (only applicable to control fields). The default value is 0 (centered, no label).

0:	AR_DVAL_IMAGE_CENTER (centered, no label).
1:	AR_DVAL_IMAGE_LEFT (to the left of the label).
2:	AR_DVAL_IMAGE_RIGHT (to the right of the label).
3:	AR_DVAL_IMAGE_ABOVE (above the label).
4:	AR_DVAL_IMAGE_BELOW (below the label).

- AR_DPROP_BUTTON_MAINTAIN_RATIO (unsigned long): A flag that indicates whether a scaled button image preserves its original width-to-height ratio (only applicable to control fields). Valid values for this property are 1 (TRUE) and 0 (FALSE). The default value is 0 (do not preserve ratio). This property only affects the image when AR_DPROP_BUTTON_SCALE_IMAGE is TRUE.
- AR_DPROP_BUTTON_SCALE_IMAGE (unsigned long): A flag that indicates whether the button image shrinks or grows to fill the space available after allocating space for the button label (only applicable to control fields). Valid values for this property are 1 (TRUE) and 0 (FALSE). The default value is **0** (leave image at original size).
- AR_DPROP_BUTTON_TEXT (character string): The button label for the field (only applicable to control fields). This label can be as many as 30 bytes in length (limited by AR_MAX_NAME_SIZE).
- AR_DPROP_CHARFIELD_AUTO_COMPLETE (unsigned long): A value that controls whether Auto Completion is in effect for a character field that has a menu attached. The default value is None.

0:	None (AR_DVAL_AUTO_COMPLETE_NONE).
1:	Leading match (AR_DVAL_AUTO_COMPLETE_LEADING).
2:	Anywhere Match (AR_DVAL_AUTO_COMPLETE_ANYWHERE).

- AR_DPROP_CHARFIELD_AUTO_COMPLETE_MATCH_BY (unsigned long): A value that determines how Auto Completion matching occurs.

0:	Match by menu value (AR_DVAL_AUTO_COMPLETE_MATCH_BY_VALUE).
1:	Match by menu label (AR_DVAL_AUTO_COMPLETE_MATCH_BY_LABEL).

- AR_DPROP_CHARFIELD_DISPLAY_TYPE (unsigned long): A value that indicates whether the character field is a drop-down list, editable field, edit/masked field, or file system browser.

0:	Editable field (AR_DVAL_CHARFIELD_EDIT)
1:	Drop-down field (AR_DVAL_CHARFIELD_DROPDOWN)
2:	Edit Masked (AR_DVAL_CHARFIELD_MASKED)
3:	File system browser (AR_DVAL_CHARFIELD_FILE)
4:	Rich text field (AR_DVAL_CHARFIELD_RICH_TEXT)
5:	Editable rich text field (AR_DVAL_CHARFIELD_RICH_TEXT_ADV)

- AR_DPROP_CNTL_TYPE (unsigned long): A bitmask that indicates the screen appearance for a control field.

Bit 0:	Button (AR_DVAL_CNTL_BUTTON).
Bit 1:	Menu item (AR_DVAL_CNTL_MENU).
Bit 2:	Toolbar button (AR_DVAL_CNTL_TOOLBAR).

- AR_DPROP_COL_HEADBKG_GRADTYPE (unsigned long): A value that enables a color fill gradient effect for a background color of the table column header.

- AR_DPROP_COL_HEADBKG_GRADCOLOR (character string): When AR_DPROP_COL_HEADBKG_GRADTYPE is set to a gradient type, this value specifies the main color in the effect.
- AR_DPROP_COL_HEADBKG_GRADBKGCOLOR (character string): When AR_DPROP_COL_HEADBKG_GRADTYPE is set to a gradient type, this value specifies the second color in the effect.
- AR_DPROP_COLOR_FILL_GRADIENT (character string): When AR_DPROP_COLOR_FILL_GRADIENT_EFFECT is set to some gradient effect, this value specifies the second color in the effect. See other color properties for value examples.
- AR_DPROP_COLOR_FILL_GRADIENT_EFFECT (unsigned long): A value that enables a color fill gradient effect for a background color of panel field. All the choices except No gradient require specification of AR_DPROP_COLOR_FILL_GRADIENT.

0:	No gradient (AR_DVAL_GRADIENT_EFFECT_NONE).
1:	Horizontal and linear (AR_DVAL_GRADIENT_EFFECT_LINEAR_HORIZONTAL).
2:	Vertical and linear (AR_DVAL_GRADIENT_EFFECT_LINEAR_VERTICAL).
3:	Horizontal and reflected (AR_DVAL_GRADIENT_EFFECT_REFLECTED_HORIZONTAL).
4:	Vertical and reflected (AR_DVAL_GRADIENT_EFFECT_REFLECTED_VERTICAL).

- AR_DPROP_COLOR_FILL_OPACITY (real): A value that controls the opacity of the background color of a panel. Specify a real number between 0 (completely transparent) and 1.0 (completely opaque). The default is 1.0.
- AR_DPROP_COLOR_GRADIENT_HEADER (character string): When AR_DPROP_COLOR_FILL_GRADIENT_EFFECT_HEADER is set to some gradient effect, this value specifies the second color in the effect. See other color properties for value examples.
- AR_DPROP_COLOR_GRADIENT_EFFECT_HEADER (unsigned long): The gradient effect, if any, for headers of panels in a collapsible or accordion panel holder. For appropriate values, see AR_DPROP_COLOR_FILL_GRADIENT_EFFECT.
- AR_DPROP_COLUMN_ORDER (integer): The order of a column in a table field. The column with the lowest value is the left-most displayed.
- AR_DPROP_COLUMN_WIDTH (integer): The width of a column in a table field.
- AR_DPROP_COORDS (ARCoordList): A set of coordinates (X,Y) that specify the vertex location (only applicable to line- and box-type trim fields). For line-type trim fields, identify the vertices of the line segment by using two coordinate pairs. For box-type trim fields, identify the corners of the box (clockwise from the upper-left corner) by using four coordinate pairs (see also AR_DPROP_BBOX). To minimize display differences across a variety of operating systems and screen resolutions, the clients standardize all coordinate values before storing them in the database and then map these values to the current environment on retrieval. When specifying (or retrieving) coordinate data for any property with ARCoordList, you must apply the same translation algorithms in order for BMC Remedy Developer Studio to display the field correctly. For additional information, see Using ARCoordList to specify field size and location.
- AR_DPROP_DATA_BBOX (ARCoordList): A set of coordinates (X,Y) that specify the screen boundaries (bounding box) for the field's data area (only applicable to data fields). For example, this

property locates the entry box for a character field and the buttons for a radio button selection field. The first coordinate pair identifies the upper-left corner for the data area. The second coordinate pair identifies the lower-right corner. The coordinates specified are relative to the field's location as specified in its `AR_DPROP_BBOX` property, not to the window or panel containing it.

- `AR_DPROP_DATA_COLS` (unsigned long): The number of text columns to display for the field (only applicable to data fields). `AR_DPROP_DATA_BBOX` overrides this value.
- `AR_DPROP_DATA_EXPAND_BBOX` (`ARCoordList`): A set of coordinates (X,Y) that specify the screen boundaries (bounding box) for the field's text editor button (only applicable to diary and long character fields). The first coordinate pair identifies the upper-left corner for the button. The second coordinate pair identifies the lower-right corner. The coordinates specified are relative to the field's location as specified in its `AR_DPROP_BBOX` property, not to the window.
- `AR_DPROP_DATA_MENU` (unsigned long): A flag that indicates whether the field has a drop-down list (only applicable to selection fields). Valid values for this property are `1` (`TRUE`) and `0` (`FALSE`). The default value is `0` (no drop-down list).
- `AR_DPROP_DATA_MENU_BBOX` (`ARCoordList`): A set of coordinates (X,Y) that specify the screen boundaries (bounding box) for the field's menu button (only applicable to character fields with an attached menu). The first coordinate pair identifies the upper-left corner for the button. The second coordinate pair identifies the lower-right corner. The coordinates specified are relative to the field's location as specified in its `AR_DPROP_BBOX` property, not to the window.
- `AR_DPROP_DATA_RADIO` (unsigned long): A flag that indicates whether the field uses radio option buttons (only applicable to selection fields). Valid values for this property are `1` (`TRUE`) and `0` (`FALSE`). The default value is `0` (does not use option buttons).
- `AR_DPROP_DATA_ROWS` (unsigned long): The number of text rows to display for the field (only applicable to data fields).
- `AR_DPROP_DATA_SPIN` (unsigned long): A flag that indicates whether the field has a numeric spinner (only applicable to integer fields). Valid values for this property are `T` (`TRUE`) and `F` (`FALSE`). The default value is `F` (no numeric spinner).
- `AR_DPROP_DATETIME_POPUP` (unsigned long): A value that indicates the display type of a date/time field. The default value is `0` (both date and time).

0:	Both date and time (AR_DVAL_DATETIME_BOTH) (default setting).
1:	Time only (AR_DVAL_DATETIME_TIME).
2:	Date only (AR_DVAL_DATETIME_DATE).

- `AR_DPROP_DEPTH_EFFECT` (unsigned long):A value that indicates the three-dimensional style for the field. This property, in conjunction with the `AR_DPROP_LINE_WIDTH` property, determines the overall look of line- and box-type trim fields.

0:	No three-dimensional effect (AR_DVAL_DEPTH_EFFECT_FLAT).
1:	Field appears to lie above screen surface (AR_DVAL_DEPTH_EFFECT_RAISED).
2:	Field appears to lie below screen surface (AR_DVAL_DEPTH_EFFECT_SUNKEN).
4:	Field appears to be etched into screen surface (AR_DVAL_DEPTH_EFFECT_ETCHED).

- AR_DPROP_DISPLAY_PARENT (unsigned long): A value that indicates the parent field that the field appears in. Specify the field ID of the parent. A panel holder field is the parent of the panel fields displayed in it, and a panel field is the parent of the fields displayed in it. A value of 0 indicates that the field's parent is the form view and not another field. The default value is 0.
- AR_DPROP_DRILL_DOWN (unsigned long): A bitmask that indicates whether drill-down is enabled for a table field. Specify bit 0 to disable and 1 to enable. When drill-down is enabled and a user double-clicks a row in the table field, a Modify window opens for the entry represented by that row.
- AR_DPROP_DROP_SHADOW (unsigned long): A value that indicates a drop shadow for panels and panel holders.

0:	No shadow (AR_DVAL_DROP_SHADOW_NONE) (default setting).
1:	Drop shadow below and to the right of the object border (AR_DVAL_DROP_SHADOW_NORTH_WEST).

- AR_DPROP_ENABLE (unsigned long): A value that specifies the enabled or disabled status of the field. For control fields, this setting applies to all specified screen appearances (see AR_DPROP_CNTL_TYPE).

1:	View and select only (data); disabled (trim or control) (AR_DVAL_ENABLE_READ_ONLY).
2:	View, select, and change (data); enabled (trim or control) (AR_DVAL_ENABLE_READ_WRITE) (default setting).
3:	Disabled (data, trim, or control) (AR_DVAL_ENABLE_DISABLE).

- AR_DPROP_ENABLE_CLEAR (unsigned long): A value that determines whether the special value "(Clear)" is added to character fields of type drop-down list. Specify AR_DVAL_ENABLE_CLEAR_ALWAYS to add the special value in all form display modes. Specify AR_DVAL_ENABLE_CLEAR_SEARCH_ONLY to remove the special value in new or modify modes, forcing users to enter a value.
- AR_DPROP_FIELD_CUSTOMSTYLE (character string): Specifies a custom CSS style for a field.
- AR_DPROP_FIELD_HIGHLIGHT (unsigned long): A flag that enables field highlighting for a field to occur when active links change the value of the field. Specify AR_DVAL_FIELD_HIGHLIGHT_DISABLE (the default) or AR_DVAL_FIELD_HIGHLIGHT_ENABLE.
- AR_DPROP_FIELD_HIGHLIGHT_END_COLOR (character string): A color value that specifies the second background color as part of the field highlighting effect, enabled by setting the AR_DPROP_FIELD_HIGHLIGHT property. See other color properties for value examples. The default is white.
- AR_DPROP_FIELD_HIGHLIGHT_START_COLOR (character string): A color value that specifies the first background color as part of the field highlighting effect, enabled by setting the AR_DPROP_FIELD_HIGHLIGHT property. See other color properties for value examples. The default is white.
- AR_DPROP_FIELD_MAX_HEIGHT (integer): When a field is positioned within a container using fill layout, this value specifies the maximum width of the field.
- AR_DPROP_FIELD_MAX_WIDTH (integer): When a field is positioned within a container using fill layout, this value specifies the maximum width of the field.

- AR_DPROP_FIELD_MIN_HEIGHT (integer): When a field is positioned within a container using fill layout, this value specifies the minimum height of the field. If the container shrinks below this value, a scroll bar will appear.
- AR_DPROP_FIELD_MIN_WIDTH (integer): When a field is positioned within a container using fill layout, this value specifies the minimum width of the field. If the container shrinks below this value, a scroll bar will appear.
- AR_DPROP_FIELD_ROUNDED (character string): This value specifies rounding of corners for panel fields using the web. To specify all corners rounded to the same radius, give a single radius value such as 10. To specify each corner's radius independently, use a comma-separated list (such as 5, 10, 5, 10) where the values are applied to the top-left, top-right, bottom-left, and bottom-right corners, respectively. By default, all corners are square.
- AR_DPROP_HIDE_PANELHOLDER_BORDERS: This value is a flag to specify that a panel holder should be borderless. Specify FALSE (0) for the default border, or TRUE (1) to suppress the border.
- AR_DPROP_HTML_TEXT (character string): The text for a text-type trim field that includes a URL. The string must include a standard HTML anchor tag (**<A>**) that encloses the text to be linked and specifies the URL to link to. Text that does not fit within its bounding box causes the form window to scale up to accommodate it (see AR_DPROP_BBOX).
- AR_DPROP_HTML_TEXT_COLOR (character string): The color of the linked text for a text-type trim field that includes a URL (see AR_DPROP_HTML_TEXT). Use one of the values listed for AR_DPROP_LABEL_COLOR_TEXT or specify a custom color as 0xBBGGRR, a string concatenating the two-digit hexadecimal values for blue, green, and red. The default value is the link color specified in the user's browser preferences.
- AR_DPROP_IMAGE (ARByteList): The icon image for a toolbar-type control field. Specify a byte list type of 1 (AR_BYTE_LIST_WIN30_BITMAP) for images in bitmap (**.bmp**) and DIB (**.dib**) format. Specify 0 (AR_BYTE_LIST_SELF_DEFINED) for all other image formats.

  Note

  The only image formats currently supported on toolbar control fields are bitmap and DIB. The only image size currently supported is 16 pixels wide by 15 pixels high. Images of other sizes are not scaled to fit. When creating images, use simple images and the 216-color palette for best results.

- AR_DPROP_JUSTIFY (unsigned long): A value that indicates the horizontal justification of text within its bounding box.

1:	Left-aligned (AR_DVAL_JUSTIFY_LEFT).
2:	Centered (AR_DVAL_JUSTIFY_CENTER).
4:	Right-aligned (AR_DVAL_JUSTIFY_RIGHT).

- AR_DPROP_LABEL (character string): The screen label for the field (only applicable to data fields). This label can be as many as 30 bytes in length (limited by AR_MAX_NAME_SIZE).
- AR_DPROP_LABEL_BBOX (ARCoordList): A set of coordinates (X,Y) that specify the screen boundaries (bounding box) for the field label (only applicable to data fields). The first coordinate pair

identifies the upper-left corner for the label. The second coordinate pair identifies the lower-right corner. The coordinates specified are relative to the field's location as specified in its `AR_DPROP_BBOX` property, not to the window or panel containing it.

- `AR_DPROP_LABEL_COLOR_TEXT` (character string): The color of the field label. Specify the color as `#BBGGRR`, a string concatenating the two-digit hexadecimal values for blue, green, and red.
- `AR_DPROP_LABEL_POS_ALIGN` (unsigned long): A value that indicates the vertical alignment of the label within the sector (see the Note that follows).

1:	Top-aligned (AR_DVAL_ALIGN_TOP).
2:	Middle-aligned (AR_DVAL_ALIGN_MIDDLE).
4:	Bottom-aligned (AR_DVAL_ALIGN_BOTTOM).

Note

The following two combinations of values for the `AR_DPROP_LABEL_POS_SECTOR`, `AR_DPROP_LABEL_POS_JUSTIFY`, and `AR_DPROP_LABEL_POS_ALIGN` properties provide the best-looking form views in BMC Remedy Mid Tier:

SECTOR	JUSTIFY	ALIGN		Label position
north	left	bottom	>	top
west	left	top	>	left

- `AR_DPROP_LABEL_POS_JUSTIFY` (unsigned long): A value that indicates the horizontal alignment of the label within the sector (see the Note that follows `AR_DPROP_LABEL_POS_ALIGN`).

1:	Left-justified (AR_DVAL_JUSTIFY_LEFT).
2:	Center-justified (AR_DVAL_JUSTIFY_CENTER).
4:	Right-justified (AR_DVAL_JUSTIFY_RIGHT).

- `AR_DPROP_LABEL_POS_SECTOR` (bitmask): A value that indicates the label position relative to the field. This property, in conjunction with the `AR_DPROP_LABEL_POS_JUSTIFY` and `AR_DPROP_LABEL_POS_ALIGN` properties, determines the specific placement and alignment of the screen label (see the Note that follows `AR_DPROP_LABEL_POS_ALIGN`).

0:	Suppress label display (AR_DVAL_SECTOR_NONE).
2:	Above the field (AR_DVAL_SECTOR_NORTH).
16:	Left of the field (AR_DVAL_SECTOR_WEST) (default setting).

- `AR_DPROP_LAYOUT_POLICY` (unsigned long): This value is used for views and fields that act like containers (panel fields) to determine how the container will arrange the layout of its child fields. Specify AR_DVAL_LAYOUT_XY if the container should honor the field's bounding box (this is the default). Specify AR_DVAL_LAYOUT_FILL to have the container automatically resize all visible children to the width of the container, resize the height of all children to be equal to the container's total height, and stack the visible children vertically on top of each other. Specify AR_DVAL_LAYOUT_FLOW for RTF fields to have the container automatically resize all visible children to the width of the container, resize the height of all children to be equal to the container's total height, and stack the visible children on top of each other.

- AR_DPROP_LINE_WIDTH (unsigned long): The line width for the field (only applicable to line- and box-type trim fields). The default value is 3 pixels.
- AR_DPROP_LOCALIZATION_REQUIRED (unsigned long): A value used to mark a field for some special purpose, with values TRUE (1) or FALSE (0). The default is FALSE. This flag was introduced to allow developers to keep track of fields that require localization. It has no effect at runtime.
- AR_DPROP_MENU_HELP (character string): The help text for a menu-type control field. This text can be as many as 30 bytes in length (limited by AR_MAX_NAME_SIZE).
- AR_DPROP_MENU_MODE (unsigned long): A value that indicates the type of menu item (only applicable to control fields).

0:	Leaf-level menu item (AR_DVAL_CNTL_ITEM).
2:	Separator item (AR_DVAL_CNTL_SEPARATOR).
5:	Top- or intermediate-level menu item (AR_DVAL_CNTL_A_MENU).

- AR_DPROP_MENU_PARENT (unsigned long): A value that specifies the parent menu item for the field (only applicable to control fields). Specify the field ID of the parent control field for leaf- or intermediate-level menu items. Specify 0 for top-level menu items.
- AR_DPROP_MENU_POS (unsigned long): A value that determines the menu position for the field (only applicable to control fields). For leaf- or intermediate-level menu items, the value associated with each field in a form view (VUI) determines the top-to-bottom order of the items in the menu (from lowest to highest, starting with one). For top-level menu items, this value determines the left-to-right order of the items in the menu bar.
- AR_DPROP_MENU_TEXT (character string): The menu item text for the field (only applicable to control fields). This text can be as many as 30 bytes in length (limited by AR_MAX_NAME_SIZE).
- AR_DPROP_NAVBAR_INITIAL_SELECTED_ITEM (field ID): Specifies that when you open a form, the navigation bar is initialized with the selected item.
- AR_DPROP_NAVBAR_WORKFLOW_ON_SELECTED_ITEM (integer): Specifies whether workflow is executed when the item is selected. Values are 1 (TRUE, execute workflow on selection) or 0 (FALSE, select the item but do not execute workflow).
- AR_DPROP_NAVBAR_SELECT_ITEM_ON_CLICK (integer): Specifies whether to move selection to an item when that item is clicked. Values are 1 (TRUE, move selection on click) or 0 (FALSE, do nothing on click).
- AR_DPROP_NONE: Indicates the NULL display property. You can specify no display properties by passing an empty property list. This property is ignored if the list also contains other display properties.

- AR_DPROP_ORIENTATION (unsigned long): A value that specifies whether child fields are arranged next to each other horizontally (AR_DVAL_ORIENTATION_HORIZONTAL) or vertically at 90 ^{0} (AR_DVAL_ORIENTATION_VERTICAL) or vertically at 270 ^{0} (AR_DVAL_ORIENTATION_VERTICAL _UP). The default is horizontal.
- AR_DPROP_PAGE_BODY_STATE (unsigned long): This value controls whether a panel within a collapsible or accordion panel holder is initially collapsed (specify AR_DVAL_PAGE_BODY_COLLAPSE) or expanded (specify AR_DVAL_PAGE_BODY_EXPAND). The default is AR_DVAL_PAGE_BODY_COLLAPSE.
- AR_DPROP_PAGE_HEADER_COLOR (character string): The background color of a header used in a panel within a collapsible, accordion, or splitter panel holder. See other color properties for value examples.
- AR_DPROP_PAGE_HEADER_STATE (unsigned long): A flag to indicate whether a header is displayed as part of a panel within a collapsible panel holder, using values AR_DVAL_PAGE_HEADER_HIDDEN or AR_DVAL_PAGE_HEADER_VISIBLE. The default is AR_DVAL_PAGE_HEADER_HIDDEN.
- AR_DPROP_PAGE_INITIAL_SIZE (integer): This value applies to a panel field that is contained by a collapsible or splitter panel holder. It controls the initial size in the dimension appropriate to the orientation--for example, in a vertical panel holder it controls the height.
- AR_DPROP_PAGE_MAX_SIZE (integer): This value applies to a panel field that is contained by a splitter panel holder. It controls the maximum size in the dimension appropriate to the orientation--for example, in a vertical panel holder it controls the maximum height. Note that as the container resizes, the penultimate panel might grow beyond this value.
- AR_DPROP_PAGE_MIN_SIZE (integer): This value applies to a panel field that is contained by a splitter panel holder. It controls the minimum size in the dimension appropriate to the orientation--for example, in a vertical panel holder it controls the minimum height.
- AR_DPROP_PAGEHOLDER_DISPLAY_TYPE (unsigned long): This value controls the style of a panel holder. The default is Tabbed.

0:	Tabbed (AR_DVAL_PAGEHOLDER_DISPLAY_TYPE_TABCTRL)
1:	Collapsible (AR_DVAL_PAGEHOLDER_DISPLAY_TYPE_STACKEDVIEW)
2:	Splitter (AR_DVAL_PAGEHOLDER_DISPLAY_TYPE_SPLITTERVIEW)
3:	Accordion (AR_DVAL_PAGEHOLDER_DISPLAY_TYPE_ACCORDION)

- AR_DPROP_PAGEHOLDER_INIT_PAGE (integer): Specifies relative offset of the panel to initially display in a panel holder. The default is 0.
- AR_DPROP_PAGEHOLDER_MARGIN_BOTTOM (integer): Specifies the bottom internal margin for panel holder. The default is 0.
- AR_DPROP_PAGEHOLDER_MARGIN_LEFT (integer): Specifies the left internal margin for panel holder. The default is 0.
- AR_DPROP_PAGEHOLDER_MARGIN_RIGHT (integer): Specifies the right internal margin for panel holder. The default is 0.
- AR_DPROP_PAGEHOLDER_MARGIN_TOP (integer): Specifies the top internal margin for panel holder. The default is 0.
- AR_DPROP_PAGEHOLDER_SPACING (integer): Specifies the internal vertical spacing between panels within a panel holder. The default is 0.

- AR_DPROP_PANELHOLDER_SPLITTER (unsigned long): A value that applies to the splitter state property of the panel holder field. It controls whether the splitter bar is visible and can be dragged to resize the panels. The splitter bar can be manipulated by the end user. The default value is AR_DVAL_SPLITTER_SHOW.

0:	Splitter bar is not visible and cannot be dragged (AR_DVAL_SPLITTER_HIDE).
1:	Splitter bar is visible and can be dragged (AR_DVAL_SPLITTER_SHOW).
2:	Splitter bar is not visible but can be dragged (AR_DVAL_SPLITTER_INVISIBLE).

- AR_DPROP_PUSH_BUTTON_IMAGE (ARByteList): The icon image for a button. Specify a byte list type of 1 (AR_BYTE_LIST_WIN30_BITMAP) for images in bitmap (**.bmp**) and DIB (**.dib**) format. Specify 2 (AR_BYTE_LIST_JPEG) for JPEG (**.jpg** or **.jpeg**) format.

Note

The only image formats currently supported on button fields are JPEG, bitmap, and DIB. There is no limit to the pixel dimensions or file size of a button image, but large images can slow down your system's performance. When creating images, use simple images and the 216-color web palette for best results.

- AR_DPROP_SORT_DIR (integer): A value that indicates the sort direction of a column in a table field. Specify 0 for ascending (AR_DVAL_SORT_DIR_ASCENDING) and 1 for descending (AR_DVAL_SORT_DIR_DESCENDING).
- AR_DPROP_SORT_SEQ (integer): The sort order of a column in a table field. The system sorts based on the column with the lowest value first. To keep from sorting based on a column, specify 0 (the default).
- AR_DPROP_TAB_ORDER (unsigned long): A value that determines the field tab order (from lowest to highest).
- AR_DPROP_TABLE_COL_DISPLAY_TYPE (unsigned long): Specifies the display type of a column field.

0:	Non-editable (AR_DVAL_TABLE_COL_DISPLAY_NONEDITABLE)
1:	Allow editing (AR_DVAL_TABLE_COL_DISPLAY_EDITABLE)
2:	Display HTML (AR_DVAL_TABLE_COL_DISPLAY_HTML)
3:	Display as a cell field in a cell-based table (AR_DVAL_TABLE_COL_DISPLAY_PAGE_DATA)

- AR_DPROP_TABLE_COLUMN_CHECKBOX (unsigned long): A value that determines whether a select/cancel all check-box column is added to a list view or results list table in a browser. If the column is added to the table, the Select All and Deselect All buttons do not appear in the table footer. For more information, see, Adding a Select All or Cancel All check-box column to tables.

0:	(Default) Do not add a select/cancel all check-box column to the table (AR_DVAL_TABLE_COLUMN_CHECKBOX_DISABLE).

1:	Add a select/cancel all check-box column to the table (`AR_DVAL_TABLE_COLUMN_CHECKBOX_ENABLE`).

- `AR_DPROP_TABLE_COLUMN_HEADER_TEXT_COLOR` (character string): The color of the text in the column header used in a table.
- `AR_DPROP_TABLE_DISPLAY_TYPE` (integer): Dictates the appearance of a table field in a browser.

0:	Standard table (AR_DVAL_TABLE_DISPLAY_TABLE).
1:	Results list table (AR_DVAL_TABLE_DISPLAY_RESULTS_LIST).
2:	Notification alert table (AR_DVAL_TABLE_DISPLAY_NOTIFICATION).
3:	Display from a single source table (AR_DVAL_TABLE_DISPLAY_SINGLE_TABLE_TREE).
4:	(reserved for future use)
5:	Display as a cell-based table (AR_DVAL_TABLE_DISPLAY_PAGE_ARRAY)

- `AR_DPROP_TABLE_HDRFTR_GRADTYPE` (unsigned long): A value that enables a color fill gradient effect for a background color of the table header and footer.
- `AR_DPROP_TABLE_HDRFTR_GRADCOLOR` (character string): When `AR_DPROP_TABLE_HDRFTR_GRADTYPE` is set to a gradient type, this value specifies the main color in the effect.
- `AR_DPROP_TABLE_HDRFTR_GRADBKGCOLOR` (character string): When `AR_DPROP_TABLE_HDRFTR_GRADTYPE` is set to a gradient type, this value specifies the second color in the effect.
- `AR_DPROP_TABLE_PAGE_ARRAY_BOTTOM_MARGIN` (integer): Specifies the bottom internal margin for a cell-based table. The default is `0`.
- `AR_DPROP_TABLE_PAGE_ARRAY_HOR_SPACE` (integer): For a cell-based table, specifies how much spacing should appear between horizontally adjacent repeating panels. The default is `0`.
- `AR_DPROP_TABLE_PAGE_ARRAY_LEFT_MARGIN` (integer): Specifies the left internal margin for a cell-based table. The default is `0`.
- `AR_DPROP_TABLE_PAGE_ARRAY_RIGHT_MARGIN` (integer): Specifies the right internal margin for a cell-based table. The default is `0`.
- `AR_DPROP_TABLE_PAGE_ARRAY_TOP_MARGIN` (integer): Specifies the top internal margin for a cell-based table. The default is `0`.
- `AR_DPROP_TABLE_PAGE_ARRAY_VER_SPACE` (integer): For a cell-based table, specifies how much spacing should appear between horizontally adjacent repeating panels. The default is `0`.
- `AR_DPROP_TABLE_PAGE_VISIBLE_COLUMNS` (integer): Specify the number of visible columns to display for a cell-based table. The default is `1`. Note that when a cell-based table is in a fill layout, the number of columns is dynamically determined based on the width of the containing field.
- `AR_DPROP_TABLE_PANEL_BBOX` (ARCoordList): This value is used when rendering cell-based tables to determine the dimensions of the repeating area.
- `AR_DPROP_TEXT` (character string): The text for a text-type trim field. Text that does not fit within its bounding box causes the form window to scale up to accommodate it (see `AR_DPROP_BBOX`).
- `AR_DPROP_TEXT_FONT_STYLE` (character string): The name of the font style to use for a text-type trim field.

Editor:	Font used for data fields.
Optional:	Font used for optional data field labels.
PushButton:	Font used for button labels.
System:	Font used for system-generated data field labels.
RadioButton:	Font used for option button choices.
Required:	Font used for required data field labels.
Header1:	Font used for titles.
Header2:	Font used for major headings.
Header3:	Font used for minor headings.
Note:	Font used for notes.
Detail:	Font used for detail information.

- `AR_DPROP_TOOLBAR_MODE` (unsigned long): A value that indicates the type of toolbar item (only applicable to control fields).

0:	Toolbar button (AR_DVAL_CNTL_ITEM).
2:	Toolbar separator (AR_DVAL_CNTL_SEPARATOR).

- `AR_DPROP_TOOLBAR_POS` (unsigned long): A value that determines the toolbar position for the field (only applicable to control fields). The value associated with each field in a form view (VUI) determines the left-to-right order of the icons in the toolbar (from lowest to highest, starting with one).
- `AR_DPROP_TOOLTIP` (character string): The tooltip text for a toolbar-type control field. This text can be as many as 30 bytes in length (limited by `AR_MAX_NAME_SIZE`).
- `AR_DPROP_TRIM_TYPE` (unsigned long): A value that indicates the type of trim field.

1:	Horizontal or vertical line (AR_DVAL_TRIM_LINE).
2:	Box (rectangle) (AR_DVAL_TRIM_SHAPE).
3:	One or more rows of text (AR_DVAL_TRIM_TEXT).

- `AR_DPROP_VERTNAV_SUBLEVELTWO_COLOR` (character string): The sub level alternate background color. This setting is applicable only to Vertical Navigation Bar in the Flyout mode.
- `AR_DPROP_VIEW_BORDER_COLOR` (character string): The color of the border of the view fields.
- `AR_DPROP_VIEWFIELD_BORDERS`: A value that indicates whether borders are hidden or shown for view fields.

0:	For BMC Remedy Mid Tier, the borders are shown based on the display (AR_DVAL_VIEWFIELD_BORDERS_DEFAULT).
1:	No borders on view fields (AR_DVAL_VIEWFIELD_BORDERS_NONE).
2:	Borders on view fields (AR_DVAL_VIEWFIELD_BORDERS_ENABLE).

- `AR_DPROP_VIEWFIELD_SCROLLBARS`: A value that indicates whether scroll bars are hidden or shown for view fields.

0:	Add scroll bars if needed. (AR_DVAL_VIEWFIELD_SCROLLBARS_AUTO).
1:	Always show scroll bars (AR_DVAL_VIEWFIELD_SCROLLBARS_ON).
2:	Hide scroll bars; content might be clipped. (AR_DVAL_VIEWFIELD_SCROLLBARS_HIDDEN).

- `AR_DPROP_VISIBLE (ENUM)`: A flag that indicates the visibility of the field. Valid values for this property are 0 (hidden) and 1 (visible). The default value is 1 (visible). For control fields, this setting applies to all specified screen appearances (see AR_DPROP_CNTL_TYPE).
- `AR_DPROP_Z_ORDER` (unsigned long): A value that determines the field drawing order. The value associated with each field in a form view (VUI) determines the back-to-front layering of the fields on the screen (from lowest to highest). The Z-order values of all fields associated with a given view must be unique. If two or more fields have duplicate Z-order values, the system orders them at random.

Note

BMC Remedy Mid Tier displays table fields, panel holder fields, and buttons with images in front of other fields regardless of their Z-order.

See also

ARCreateVUI, ARDeleteField, ARGetField, ARGetMultipleFields, ARGetListField, ARSetField. See FreeAR for: FreeARFieldLimitList, FreeARPermissionList, FreeARStatusList, FreeARValueStruct.

6.2.9 ARCreateFilter

Description

Creates a new filter with the indicated name on the specified server. The filter takes effect immediately and remains in effect until changed or deleted.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateFilter(
   ARControlStruct *control,
   ARNameType name,
   unsigned int order,
   ARWorkflowConnectStruct *schemaList,
   unsigned int opSet,
   unsigned int enable,
   ARQualifierStruct *query,
   ARFilterActionList *actionList,
   ARFilterActionList *elseList,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   unsigned int errorFilterOptions
   ARNameType errorFilterName
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARCreate*` function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the filter to create. The names of all filters on a given server must be unique.

order

A value between `0` and `1000` (inclusive) that determines the filter execution order. When multiple filters are associated with a form, the value associated with each filter determines the order in which they are processed (from lowest to highest).

schemaList

The list of form names the filter is linked to. The filter must be associated with a single form or a list of forms that currently exist on the server.

opSet

A bitmask that indicates the operations that trigger the filter.

Bit 0:	Execute the filter on get form operations (`AR_OPERATION_GET`).
Bit 1:	Execute the filter on set form operations (`AR_OPERATION_SET`).

Bit 2:	Execute the filter on create form operations (AR_OPERATION_CREATE).
Bit 3:	Execute the filter on delete form operations (AR_OPERATION_DELETE).
Bit 4:	Execute the filter on merge form operations (AR_OPERATION_MERGE).
Bit 5:	Execute the filter from a filter guide (AR_OPERATION_GUIDE).
Bit 6:	Execute the filter on a Service active link action operation (AR_OPERATION_SERVICE).

enable

A flag to enable or disable this filter. A value of 0 disables the filter, causing its condition checks and associated actions to not be performed. A value of 1 enables the filter, causing its conditions to be checked for each form operation specified by the opSet parameter.

query

A qualification that determines whether the filter is executed. Specify NULL or assign an operation value of 0 (AR_COND_OP_NONE) to execute the filter unconditionally.

actionList

The set of actions performed if the condition defined by the query parameter is satisfied. You can specify from 1 to 25 actions in this list (limited by AR_MAX_ACTIONS).

elseList

The set of actions performed if the condition defined by the query parameter is not satisfied. You can specify from 0 to 25 actions in this list (limited by AR_MAX_ACTIONS). Specify NULL for this parameter (or zero actions) if you do not want to define any else actions.

helpText

The help text associated with the filter. This text can be of any length. Specify NULL for this parameter if you do not want to associate help text with this object.

owner

The owner for the filter. The owner defaults to the user performing the operation if you specify NULL for this parameter.

changeDiary

The initial change diary associated with the filter. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify NULL for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to NULL, a properties list with zero properties is associated with the object, and when the filter definition is retrieved, zero properties are returned. See Server object properties.

errorFilterOptions

The error handler options for the filter. Set to `AR_FILTER_ERRHANDLER_ENABLE` to enable an error handler filter. Set to zero to disable.

errorFilterName

The name of error handler filter for this filter; `NULL` if none. See Error processing for a description of error handlers.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

`Rename` and `Delete` operations typically change multiple objects in addition to their primary target object. The `Rename` or `Delete` function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDeleteFilter, ARGetFilter, ARGetListFilter, ARSetFilter. See FreeAR for: FreeARFilterActionList, FreeARPropList, FreeARQualifierStruct, FreeARStatusList.

6.2.10 ARCreateImage

Description

Creates a new image with the indicated name on the specified server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateImage(
   ARControlStruct *control,
   ARNameType name,
   ARImageDataStruct *imageBuf,
   char *imageType,
   char *description,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the ARCreate* function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP variable of the ARSetSessionConfiguration function (see ARGetSessionConfiguration).

name

The name of the image to create. The names of all images on a given server must be unique.

imageBuf

The data for the new image. See Images and structures.

imageType

The image encoding type stored as a character string. Valid values are: `BMP`, `GIF`, `JPEG` or `JPG`, and `PNG`.

description

The description for this image. It can be as many as `2000` characters long or `NULL`.

helpText

The help text associated with the image. This text can be of any length. Specify `NULL` for this parameter if you do not want to associate help text with this object.

owner

The owner for the image. This parameter is required.

changeDiary

The initial change diary associated with the image. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a properties list with zero properties is associated with the object, and when the image definition is retrieved, zero properties are returned. See Server object properties and structures.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

`Rename` and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDeleteImage, ARGetImage, ARGetListImage, ARGetMultipleImages, ARSetImage. See FreeAR for: `FreeARImageDataStruct`, `FreeARStatusList`.

6.2.11 ARCreateLicense

Description

Adds an entry to the Add or Remove Licenses form for the current server.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"
```

```
int ARCreateLicense(
   ARControlStruct *control,
   ARLicenseInfoStruct *licenseInfo,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

licenseInfo

New license entry information, such as license key, license type, expiration date, number of licenses, and qualifier. This argument accepts other information, such as issue date, but it does not include the information in the new entry, and the extraneous information is not saved.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListLicense, ARDeleteLicense, ARValidateLicense, ARValidateMultipleLicenses. See FreeAR for: `FreeARStatusList`, `FreeARLicenseInfoStruct`.

6.2.12 ARCreateMultipleFields

Description

Creates multiple fields in a form. A call to this function produces the same result as a sequence of ARCreateField calls to create the individual fields, but it can be more efficient. Using this function requires only one call from the client to the BMC Remedy AR System server. The server can perform multiple database operations in a single transaction and avoid repeating operations such as those performed at the end of each individual ARCreateField call.

If an error occurs while creating an individual field, no fields are created.

The list input arguments define the fields that the server creates. Each list type consists of a count and a pointer to an array of values. If all values in a list are `NULL`, the argument can be `NULL` or a zero-length list. All list counts that are not zero must be equal.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateMultipleFields(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalIdList *fieldIdList,
   ARBooleanList *reservedIdOKList
   ARNamePtrList *fieldNameList,
   ARFieldMappingList *fieldMapList,
   ARUnsignedIntList *dataTypeList,
   ARUnsignedIntList *optionList,
   ARUnsignedIntList *createModeList,
   ARUnsignedIntList *fieldOptionList,
   ARValuePtrList *defaultValList,
   ARPermissionListPtrList *permissionListList,
   ARFieldLimitPtrList *limitList,
   ARDisplayInstanceListPtrList *dInstanceListList,
   ARTextStringList *helpTextList,
   ARAccessNamePtrList *ownerList,
   ARTextStringList *changeDiaryList,
   ARPropListList *objPropListList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARCreate*` function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

schema

The name of an existing form on the server. The fields are created on that form.

fieldIdList

A list of the internal IDs of the fields to create. The IDs of all fields and form views (VUIs) associated with a given form must be unique. If an element of this list is zero, the server generates and returns the ID for the corresponding field. Otherwise, specify a value between `536868912` and `2147483647` (limited by `AR_MAX_RESERVED_FIELD_ID` in **arstruct.h**). If you specify a reserved ID, the system generates an error unless the corresponding element in the `reservedIdOKList` parameter is set to `1` (`TRUE`). (For more information, see Reserved field ranges.)

reservedIdOKList

A list of flags that specify whether the server should create the corresponding field with a reserved ID. Specify `1` (`TRUE`) for any element of this parameter to allow a reserved ID for the corresponding field. Specify `0` (`FALSE`) if you want the system to generate an error when you specify a reserved ID for the corresponding field.

fieldNameList

A list of pointers to the names of the fields to create. The names of all fields and VUIs associated with a given form must be unique. The system uses this name when it creates the SQL view of the form for report writing purposes. The field name is used to identify a field in a workflow object more easily.

The field name is different from the field label in that it is specific to a form, but not specific to a view of the form. See dInstanceListList for information about how to specify a field label. If you want the server to create a default name for a field based on the field ID, specify a `NULL` field name pointer in the corresponding location in this list.

Specify `NULL` for this parameter if you want the system to create default names for all the fields.

fieldMapList

A list of structures that specify mappings to the underlying form in which to create the fields. To create fields in a base form, specify a field type of `1` (`AR_FIELD_REGULAR`) in each structure in the list. Otherwise, specify a field type of `2`(`AR_FIELD_JOIN`), and identify the member form (primary or secondary) and field ID for the new field. If the member form is also a join form, create field mappings in all nested join forms until you can map the field to an underlying base form. See Mapping fields in schemas.

dataTypeList

A list of the data types of the fields to create.

2	`AR_DATA_TYPE_INTEGER`	Integer
3	`AR_DATA_TYPE_REAL`	Real
4	`AR_DATA_TYPE_CHAR`	Character
5	`AR_DATA_TYPE_DIARY`	Diary
6	`AR_DATA_TYPE_ENUM`	Selection
7	`AR_DATA_TYPE_TIME`	Date/time
`10`	`AR_DATA_TYPE_DECIMAL`	

		Fixed-point decimal. Values must be specified in C locale, for example 1234.56
11	AR_DATA_TYPE_ATTACH	Attachment
12	AR_DATA_TYPE_CURRENCY	Currency
31	AR_DATA_TYPE_TRIM	Trim
32	AR_DATA_TYPE_CONTROL	Control
33	AR_DATA_TYPE_TABLE	Table
34	AR_DATA_TYPE_COLUMN	Column
35	AR_DATA_TYPE_PAGE	Panel
36	AR_DATA_TYPE_PAGE_HOLDER	Panel holder

optionList

A list of flags that specify whether users must enter a value in the field.

1:	Required (data fields only) (AR_FIELD_OPTION_REQUIRED).
2:	Optional (data fields only) (AR_FIELD_OPTION_OPTIONAL).
4:	Display-only (data fields only). Works like an optional field but doesn't write to the database (AR_FIELD_OPTION_DISPLAY).

createModeList

A list of flags that specify the permission status for the field when users submit entries. The list entry is ignored for display-only fields.

1:	Any user (including guest users) can enter data in the field when submitting (AR_FIELD_OPEN_AT_CREATE).
2:	Only users who have been granted permission can enter data in the field when submitting (AR_FIELD_PROTECTED_AT_CREATE).

fieldOptionList

A list of bitmasks that specify whether the field is to be audited or copied when other fields are audited.

Bit 0:	Audit this field. (AR_FIELD_BITOPTION_AUDIT)
Bit 1:	Copy this field when other fields in the form are audited. (AR_FIELD_BITOPTION_COPY)
Bit 2:	Indicates this field is for Log Key 1. (AR_FIELD_BITOPTION_LOG_KEY1)
Bit 3:	Indicates this field is for Log Key 2. (AR_FIELD_BITOPTION_LOG_KEY2)
Bits 2 and 3:	Indicates this field is for Log Key 3. (AR_FIELD_BITOPTION_LOG_KEY3)

defaultValList

A list of pointers to the values to apply to the corresponding data field when a user submits an entry with no field value. The default value can be as many as 255 bytes in length (limited by `AR_MAX_DEFAULT_SIZE`) and must be of the same data type as the field. Put `NULL` (or `AR_DEFAULT_VALUE_NONE`) in the list for trim, control, panel, and panel holder fields or if you do not want to define a default value for a data field.

Specify `NULL` for this parameter if you do not want to specify default values for all fields.

permissionsListList

A list of pointers to lists of zero or more groups who can access the corresponding field. Specify `NULL` or an empty permission list for a field to make it accessible by users with administrator capability only. Specifying group ID `0` (Public) provides access to all users. The permission value that you assign for each group determines whether users in that group can modify the field. See Field permission values for.

Specify `NULL` for this parameter to make all fields accessible by users with administrator capability only.

limitList

A list of pointers to the value limits for the corresponding field and other properties specific to the field's type. See Defining field limits for a description of the contained structure that applies to the type of field that you are creating. The limits and properties that you define must be of the same data type as the field. Specify `NULL` (or `AR_FIELD_LIMIT_NONE`) for trim and control fields or if you do not want to define any limits or properties for the corresponding field.

Specify `NULL` for this parameter if you do not want to specify limits for any field.

dInstanceListList

A list of pointers to lists of zero or more display properties to associate with the corresponding field. You can define both display properties common to all form views (VUIs) and display properties specific to particular views. The system includes the field in each view that you specify, regardless of whether that you define any display properties for those views. If you do not specify a property for a particular view, the system uses the default value (if defined). Specify a `NULL` display instance list pointer in the list if you do not want to define any display properties for the corresponding field.

Specify `NULL` for this parameter if you do not want to specify a display instance list for any field.

helpTextList

A list of pointers to the help text associated with the corresponding field. The text can be of any length. Specify a `NULL` pointer in the list if you do not want to specify help text for the corresponding field.

Specify `NULL` for this parameter if you do not want to specify help text for any field.

ownerList

A list of pointers to the owner for the corresponding field. Specify a `NULL` pointer in the list if you want to default the owner for the corresponding field to the user performing the operation.

Specify `NULL` for this parameter if you want to default the owner to the user performing the operation for all fields.

changeDiaryList

A list of pointers to the change diary text associated with the corresponding field. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify a `NULL` pointer in the list if you do not want to specify change diary text for the corresponding field.

Specify `NULL` for this parameter if you do not want to specify any change diary text for any field.

objPropListList

A list of server object property lists that correspond to the fields in `fieldIdList`. If this parameter is set to `NULL`, a list with zero properties is associated with each object, and zero properties are returned when an `ARGetField` is performed. See Server object properties.

Return values

fieldIdList

See fieldIdList.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. This function returns error code `488` (`AR_ERROR_CREATE_MULT_FIELD_POSITION`) when an error occurred in the creation of an individual field. The zero-based position of that field in the list parameters is returned in the appended text of the status item in ASCII character format. Any additional status items refer to the field creation error that occurred. If an error occurs creating an individual field, no fields are created. For a description of all possible values, see Error checking.

See also

ARCreateField, ARSetMultipleFields. See FreeAR for: `FreeARAccessNamePtrList`, `FreeARBooleanList`, `FreeARDisplayInstanceListPtrList`, `FreeARFieldLimitPtrList`, `FreeARFieldMappingPtrList`, `FreeARInternalIdList`, `FreeARNamePtrList`, `FreeARPermissionListPtrList`, `FreeARStatusList`, `FreeARTextStringList`, `FreeARUnsignedIntList`, `FreeARValuePtrList`.

6.2.13 ARCreateOverlay

Description

Creates an overlay for a specified object, and sets the following information:

- Name of the overlay object.

- Type of overlaid object (active link, form, menu, and so on)
- (Views and fields only) ID of the overlay object
- (Views and fields only) Name of the form associated with the overlaid object
- Overlay characteristics (from the `inheritMask` and `extendMask` object properties of the `AROverlaidStruct` parameter)

For more information about overlays, see About overlays.

Privileges

BMC Remedy AR System administrator

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateOverlay(
   ARControlStruct *control
   AROverlaidStruct *overlaid
   Char objectModificationLogLabel
   ARNameType name
   ARInternalID *overlayID
   ARStatusList *status
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **server**, and **overlayGroup** fields are required.

Note

To specify the overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_ OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). If that variable is set to a valid overlay group, this function creates an overlay object that belongs to that group.

overlaid

Information about the object to be overlaid, including its type (active link, form, menu, and so on).

objectModificationLogLabel

Version control label associated with an object in the object modification log. For more information, see Labeling a collection of objects.

Return values

name

Real name of the overlay object created by this function. The name is composed of the name of the overlaid object plus the **suffix__o**:
overlaidObjectName**__o**

See Overlay object names.

overlayID

ID number of the overlay object if the overlaid object is a form view or a field.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. See Error checking for a description of all possible values.

See also

ARCreateOverlayFromObject

6.2.14 ARCreateOverlayFromObject

Description

Performs the following operations:

- Converts an object into an overlay of another origin object
  The object must be one of the following types:
  - Active link
  - Active link guide
  - Escalation
  - Filter
  - Filter guide
  - Image
  - Local application
  - Menu
  - Packing list
  - Web service
    This function works only for objects whose customizations do not violate the BMC customization guidelines.
    This function does not convert forms, form views, and fields into overlays.

- Converts an object into a custom object
  This function converts an origin object to a custom object. (For a list of objects that can be overlaid, see Working with overlays and custom objects.)
  To convert views and fields from origin to custom, the associated form must be an overlay or a custom object.
- Converts a custom object into an origin object
  This function converts a custom object into an origin object.
  To convert views and fields from custom to origin, the associated form must be an overlay. Such conversion is not be supported on custom forms.

> **Note**
>
> When a form is converted *to* origin or custom, the server implicitly converts its views and fields. However, if required, you need to explicitly convert the associated objects (for example, workflows, menus, and so on).

The Best Practice Conversion utility uses this function.

Privileges

BMC Remedy AR System administrator

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateOverlayFromObject(
   ARControlStruct *control
   AROverlaidStruct *overlaid
   AROverlaidStruct *customObj
   Char *objectModificationLogLabel
   ARNameType name
   ARInternalID *overlayID
   ARStatusList *status
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **server**, and **overlayGroup** fields are required.

Note

The overlay group must be set to 1. To specify the overlay group, use the AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP variable of the ARSetSessionConfiguration function (see ARSetSessionConfiguration). If that variable is set to a valid overlay group, this function creates an overlay or custom object that belongs to that group.

overlaid

Information about the object to be overlaid, including its type (see the following table). To convert an object to a custom object, enter NULL in this parameter.

When an existing object is turned into an overlay, all grains of the overlay are overwritten — nothing is extended or inherited. As such, the inheritMask and extendMask for this call must be set to 0.

Values for object types that can be overlaid

Value	Object type
1	AR_STRUCT_ITEM_SCHEMA
5	AR_STRUCT_ITEM_FILTER
6	AR_STRUCT_ITEM_ACTIVE_LINK
8	AR_STRUCT_ITEM_CHAR_MENU
9	AR_STRUCT_ITEM_ESCALATION
12	AR_STRUCT_ITEM_CONTAINER
14	AR_STRUCT_ITEM_VUI
15	AR_STRUCT_ITEM_FIELD
16	AR_STRUCT_ITEM_APP
17	AR_STRUCT_ITEM_IMAGE

For more information about this structure, see Overlays and structures.

customObj

Information about the object to be converted into a custom object.

objectModificationLogLabel

Version control label associated with an object in the object modification log. See Labeling a collection of objects.

Return values

name

Real name of the overlay object created by this function:

- If an object is converted into an overlay, the name of the new overlay object is composed of the name of the overlaid object plus the __**o** suffix:
  ***overlaidObjectName*__o**
  See Overlay object names*.*
- If an object is converted into a custom object, the name of the new custom object is returned, which is the same as the origin object.
  See Working with overlays and custom objects*.*

overlayID

ID number of the overlay object if the overlaid object is a view or a field. See Overlay object names.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. See Error checking for a description of all possible values.

See also

ARCreateOverlay

6.2.15 ARCreateSchema

Description

Creates a new form with the indicated name on the specified server. The nine required core fields are automatically associated with the new form.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateSchema(
   ARControlStruct *control,
   ARNameType name,
   ARCompoundSchema *schema,
   ARSchemaInheritanceList *schemaInheritanceList,
   ARPermissionList *groupList,
   ARInternalIdList *admingrpList,
   AREntryListFieldList *getListFields,
   ARSortList *sortList,
   ARIndexList *indexList,
   ARArchiveInfoStruct *archiveInfo,
   ARAuditInfoStruct *auditInfo,
   ARNameType defaultVui,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARCreate*` function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the form to create. The names of all forms on a given server must be unique.

schema

The type of form to create. The information contained in this definition depends on the form type that you specify.

`1:`	Indicates a base form (`AR_SCHEMA_REGULAR`). All other structure information is ignored if you specify this type.
`2:`	Indicates a join form (`AR_SCHEMA_JOIN`). If you specify this type, you must identify the underlying member forms and how to join them.
`3:`	Indicates a view form (`AR_SCHEMA_VIEW`). You must identify the underlying table and key field names if you specify this type.
`4:`	Indicates a display-only form (`AR_SCHEMA_DIALOG`). The `getListFields`, `sortList`, and `indexList` parameters are ignored if you specify this type.
`5:`	Indicates a vendor form (`AR_SCHEMA_VENDOR`). You must identify the underlying vendor and table names if you specify this type.

schemaInheritanceList

This is reserved for future use. Specify **NULL**.

groupList

A list of zero or more groups who can access this form. Specify an empty group list to define a form accessible only by users with administrator capability. Specify group ID `0` (Public) to provide access to all users. The permission value that you assign for each group determines whether users in that group see the form in the form list:

`1:`	Users see the form in the form list (`AR_PERMISSIONS_VISIBLE`).
`2:`	Users do not see the form in the form list (`AR_PERMISSIONS_HIDDEN`).

admingrpList

A list of zero or more groups who can administer this form (and the associated filters, escalations, and active links). Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify an empty administrator group list to define a form that can be administered only by users with administrator capability. Specify group ID `0` (Public) to provide administrator capability to all members of the Subadministrator group.

getListFields

A list of zero or more fields that identifies the default query list data for retrieving form entries. The list can include any data fields except diary fields and long character fields. The combined length of all specified fields, including separator characters, can be as many as 128 bytes (limited by `AR_MAX_SDESC_SIZE`). The query list displays the Short-Description core field if you specify `NULL` for this parameter (or zero fields). Specifying a `getListFields` argument when calling the `ARGetListEntry` function overrides the default query list data.

sortList

A list of zero or more fields that identifies the default sort order for retrieving form entries. Specifying a `sortList` argument when calling the `ARGetListEntry` function overrides the default sort order.

indexList

The set of zero or more indexes to create for the form. You can specify from 1 to 16 fields for each index (limited by `AR_MAX_INDEX_FIELDS`). Diary fields and character fields larger than 255 bytes cannot be indexed.

archiveInfo

If a form is to be archived, this is the archive information for the form. Specify `NULL` for this parameter if you do not want to create or set the archive information. These are the values for archive type:

`0:`	Deletes the archive settings for the selected form. The selected form is not archived (`AR_ARCHIVE_NONE`).
`1:`	Entries on the selected form are copied to the archive form (`AR_ARCHIVE_FORM`).

`2:`	Entries on the selected form are deleted from the source form (`AR_ARCHIVE_DELETE`).
`32:`	Attachment fields are not copied to the archive form. (`AR_ARCHIVE_NO_ATTACHMENTS`).
`64:`	Diary fields are not copied to the archive form. (`AR_ARCHIVE_NO_DIARY`).

In addition to the archive type, `archiveInfo` also stores the form name (if archive type is `AR_ARCHIVE_FORM`), time to trigger the archive, and the archive qualification criteria. For an archive form, only the name of the source form is maintained, and none of this archive information can be set.

auditInfo

If a form is to be audited, this is the audit information for the form. Specify `NULL` for this parameter if you do not want to create or set the audit information. These are the values for audit type:

`0:`	The selected form is not to be audited. (`AR_AUDIT_NONE`).
`1:`	Audit fields and copy fields on the selected form are copied to the audit shadow form (`AR_AUDIT_COPY`).
`2:`	Audit fields and copy fields on the selected form are copied to the audit log form (`AR_AUDIT_LOG`).

In addition to the audit type, `auditInfo` also stores the form name (if audit type is `AR_AUDIT_COPY` or `AR_AUDIT_LOG`) and the audit qualification criteria.

defaultVui

The label for the default view.

helpText

The help text associated with the form. This text can be of any length. Specify `NULL` for this parameter if you do not want to associate help text with this object.

owner

The owner for the form. The owner defaults to the user performing the operation if you specify `NULL` for this parameter.

changeDiary

The initial change diary associated with the form. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a properties list with zero properties is associated with the form, and a list of zero properties is returned when an `ARGetSchema` is performed. See Server object properties.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

`Rename` and `Delete` operations typically change multiple objects in addition to their primary target object. The `Rename` or `Delete` function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateField, ARDeleteSchema, ARGetSchema, ARGetListEntry, ARGetListAlertUser, ARSetField, ARSetSchema. See FreeAR for: `FreeARCompoundSchema`, `FreeAREntryListFieldList`, `FreeARIndexList`, `FreeARInternalIdList`, `FreeARPermissionList`, `FreeARPropList`, `FreeARSortList`, `FreeARStatusList`.

6.2.16 ARCreateSupportFile

Description

Creates a file that clients can retrieve by using the BMC Remedy AR System. Such files are commonly used for reports (to store them separately from the active link that calls them, preventing large downloads of unneeded information), but this function can store any file on the server. Each support file is associated with a server object.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateSupportFile(
   ARControlStruct *control,
   unsigned int fileType,
   ARNameType name,
   ARInternalId id2,
   ARInternalId fileId,
   FILE *filePtr,
   ARStatusList *status)
```

Input arguments

control

The control record for the operations. It contains information about the user requesting the operation, where the operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

fileType

The numerical value for the type of file, and the type of object the file is related to. Specify `1` (`AR_SUPPORT_FILE_EXTERNAL_REPORT`) for an external report file associated with an active link.

name

The name of the object the file is associated with, usually a form.

id2

The ID of the field or VUI, if the object is a form. If the object is not a form, set this parameter to `0`.

fileId

The unique identifier of a file within its object.

filePtr

A pointer to the support file to be created in the system. If using Windows, you must open the file in binary mode.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent, ARDeleteActiveLink, ARDeleteSupportFile, ARGetActiveLink, ARGetListSupportFile, ARGetSupportFile, ARSetActiveLink, ARSetSupportFile.

6.2.17 ARCreateVUI

Description

Creates a new form view (VUI) with the indicated name on the specified server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARCreateVUI(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalId *vuiId,
   ARNameType vuiName,
   ARLocaleType locale,
   unsigned int vuiType,
   ARPropList *dPropList,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARCreate*` function creates a custom object that belongs to that group. If no group is specified, the function creates an origin object. To specify an overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

schema

The name of the form the VUI is linked to. The VUI must be associated with a single form that currently exists on the server.

vuiId

The internal ID of the VUI to create. The IDs of all VUIs and fields associated with a given form must be unique. Specify `0` for this parameter if you want the system to generate and return the ID. Otherwise, specify a value between `536868912` and `2147483647` (limited by `AR_MAX_RESERVED_FIELD_ID` in **arstruct.h**).

vuiName

The name of the VUI to create. The names of all VUIs and fields associated with a given form must be unique.

locale

The locale of the VUI.

vuiType

The type of VUI. Specify `NULL` for this parameter if you do not want to retrieve the VUI type.

`0:`	No VUI type (`AR_VUI_TYPE_NONE`).
`1:`	Windows type (`AR_VUI_TYPE_WINDOWS`) *** -** fields in BMC Remedy User.
`2:`	Web relative type (`AR_VUI_TYPE_WEB`) - fields in view can be adjusted.
`3:`	Web absolute type (`AR_VUI_TYPE_WEB_ABS_POS`) - fields in view are fixed.

Fields marked with a * are available for legacy environments that use BMC Remedy User, which is no longer shipped with BMC Remedy AR System.

dPropList

A list of zero or more display properties to associate with the VUI (see the Display Properties section that follows). Specify `NULL` for this parameter if you do not want to define any display properties.

helpText

The help text associated with the VUI. This text can be of any length. Specify `NULL` for this parameter if you do not want to associate help text with this object.

owner

The owner for the VUI. The owner defaults to the user performing the operation if you specify `NULL` for this parameter.

changeDiary

The initial change diary associated with the VUI. This text can be of any length. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a properties list with zero properties is associated with the form, and a list of zero properties is returned when an `ARGetVUI` is performed. See Server object properties.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

Display properties

- `AR_DPROP_ALIAS_ABBREV_PLURAL` (character string): A very short, plural form of the name for the logical items represented by entries in the form this view is associated with (see `AR_DPROP_ALIAS_SINGULAR`).
- `AR_DPROP_ALIAS_ABBREV_SINGULAR` (character string): A very short, singular form of the name for the logical items represented by entries in the form this view is associated with (see `AR_DPROP_ALIAS_SINGULAR`).
- `AR_DPROP_ALIAS_PLURAL` (character string): The plural form of the name for the logical items represented by entries in the form this view is associated with (see `AR_DPROP_ALIAS_SINGULAR`).
- `AR_DPROP_ALIAS_SHORT_PLURAL` (character string): A short, plural form of the name for the logical items represented by entries in the form this view is associated with (see `AR_DPROP_ALIAS_SINGULAR`).
- `AR_DPROP_ALIAS_SHORT_SINGULAR` (character string): A short, singular form of the name for the logical items represented by entries in the form this view is associated with (see `AR_DPROP_ALIAS_SINGULAR`).

- `AR_DPROP_ALIAS_SINGULAR` (character string): The singular form of the name for the logical items represented by entries in the form this view is associated with (such as help desk calls). BMC Remedy User displays this name instead of the default "request" when referring to form entries.
- `AR_DPROP_DETAIL_BANNER_VISIBILITY` (unsigned long): A value that indicates whether the details pane is visible. Specify `0` for invisible, `1` for visible.
- `AR_DPROP_DETAIL_PANE_COLOR` (character string): The background color of the details pane. Specify the color as `#BBGGRR`, a string concatenating the two-digit hexadecimal values for blue, green, and red.
- `AR_DPROP_DETAIL_PANE_HEIGHT` (integer): The height of the details pane, in form coordinates.
- `AR_DPROP_DETAIL_PANE_IMAGE` (`ARByteList`): The background image for the details pane. Specify a byte list type of `1` (`AR_BYTE_LIST_WIN30_BITMAP`) for images in bitmap (**.bmp**) and DIB (**.dib**) format. Specify `2` (`AR_BYTE_LIST_JPEG`) for JPEG (**.jpg** or **.jpeg**) format (see `AR_DPROP_PUSH_BUTTON_IMAGE`).
- `AR_DPROP_DETAIL_PANE_WIDTH` (integer): The width of the details pane, in form coordinates. Negative values are reserved for future use.
- `AR_DPROP_IMAGE_ALIGN` (unsigned long): A value that indicates the vertical alignment of the background image for the details pane.

`1:`	Top-aligned (`AR_DVAL_ALIGN_TOP`).
`2:`	Centered (`AR_DVAL_ALIGN_MIDDLE`) (default setting).
`3:`	Expand to fill (`AR_DVAL_ALIGN_FILL`).
`4:`	Bottom-aligned (`AR_DVAL_ALIGN_BOTTOM`).
`5:`	Tile to fill (`AR_DVAL_ALIGN_TILE`).

- `AR_DPROP_IMAGE_JUSTIFY` (unsigned long): A value that indicates the horizontal justification of the background image for the details pane.

`1:`	Left-aligned (AR_DVAL_JUSTIFY_LEFT).
`2:`	Centered (AR_DVAL_JUSTIFY_CENTER).
`3:`	Expand to fill (AR_DVAL_JUSTIFY_FILL).
`4:`	Right-aligned (AR_DVAL_JUSTIFY_RIGHT).
`5:`	Tile to fill (AR_DVAL_JUSTIFY_TILE).

- `AR_DPROP_LAYOUT_POLICY` (unsigned long): This value is used for views to determine how child fields will be arranged. Specify `AR_DVAL_LAYOUT_XY` if the container should honor the fields bounding box (this is the default). Specify `AR_DVAL_LAYOUT_FILL` to have the container automatically resize all visible children to the width of the container, resize the height of all children to be equal to the container's total height, and stack the visible children vertically on top of each other.
- `AR_DPROP_PANE_LAYOUT` (integer): A value that indicates how the panes of the VUI are arranged. This property also specifies whether the layout is locked to prevent the user from changing it.

`-1:`	Results pane left of details, prompt pane on top, locked.

`-2:`	Details pane left of results, prompt pane on top, locked.
`-3:`	Results pane above details, prompt pane on top, locked.
`-4:`	Details pane above results, prompt pane on top, locked.
`-5:`	Results pane left of details, prompt pane on bottom, locked.
`-6:`	Details pane left of results, prompt pane on bottom, locked.
`-7:`	Results pane above details, prompt pane on bottom, locked.
`-8:`	Details pane above results, prompt pane on bottom, locked.
`1:`	Results pane left of details, prompt pane on top, unlocked.
`2:`	Details pane left of results, prompt pane on top, unlocked.
`3:`	Results pane above details, prompt pane on top, unlocked.
`4:`	Details pane above results, prompt pane on top, unlocked.
`5:`	Results pane left of details, prompt pane on bottom, unlocked.
`6:`	Details pane left of results, prompt pane on bottom, unlocked.
`7:`	Results pane above details, prompt pane on bottom, unlocked.
`8:`	Details pane above results, prompt pane on bottom, unlocked.

- `AR_DPROP_MENU_ACCESS` (character string): A list of client interface items an administrator can control. It is encoded in the same format as qualifications in **.arf** files. This is not supported for customer use.
- `AR_DPROP_NAMED_SEARCHES` (character string): A list of administrator-defined searches. It is encoded in the same format as qualifications in **.arf** files. This is not supported for customer use.
- `AR_DPROP_QUERY_LIST_COLOR` (character string): The name of the field whose value for a given entry controls the display color of that entry in a query results list, and a list of field values and their corresponding colors. This is not supported for customer use.
- `AR_DPROP_RESULT_BANNER_VISIBILITY`(unsigned long): A value that indicates whether the results pane is visible. Specify `0` for invisible, `1` for visible.
- `AR_DPROP_TITLE_BAR_ICON_IMAGE` (`ARByteList`): The icon image for the title bar. Specify a byte list type of `1` (`AR_BYTE_LIST_WIN30_BITMAP`) for images in bitmap (**.bmp**) and DIB (**.dib**) format. Specify `2` (`AR_BYTE_LIST_JPEG`) for JPEG (**.jpg** or **.jpeg**) format.
- `AR_DPROP_VIEW_RTL` (unsigned long): A flag that indicates a view to be rendered from right-to-left (web only). Valid values are `FALSE` (`0`) or `TRUE` (`1`).
- `AR_DPROP_VUI_DEFAULT` (unsigned long): A flag that specifies the default view (VUI) for the form. A value of `1` identifies the default VUI (with all other VUIs having a value of `0`). The system uses the first VUI returned by the `ARGetListVUI` function if you do not specify a default view.

See also

ARCreateField, ARDeleteVUI, ARGetVUI, ARGetListVUI, ARSetVUI. See FreeAR for: `FreeARPropList`, `FreeARStatusList`.

6.2.18 ARDateToJulianDate

Description

Converts a year, month, and day value to a Julian date. The Julian date is the number of days since noon, Universal Time, on January 1, 4713 BCE (on the Julian calendar). The changeover from the Julian calendar to the Gregorian calendar occurred in October, 1582. The Julian calendar is used for dates on or before October 4, 1582. The Gregorian calendar is used for dates on or after October 15, 1582.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDateToJulianDate(
   ARControlStruct *control,
   ARDateStruct *date,
   int *jd,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

date

The date structure holding the year, month, day value to convert.

Return values

jd

The resulting Julian date value.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARJulianDateToDate.

6.2.19 ARDecodeAlertMessage

Description

Decodes a formatted alert message and returns the component parts of the message to the alert client.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDecodeAlertMessage(
   ARControlStruct *control,
   unsigned char *message,
   unsigned int messageLen,
   ARTimestamp *timestamp,
   unsigned int *sourceType,
   unsigned int *priority,
   char **alertText,
   char **sourceTag,
   char **serverName,
   char **serverAddr,
   char **formName,
   char **objectId,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

message

An alert message buffer received by the client via its sockets communication with the server. The buffer is not null-terminated.
The method for receiving the message from the socket is as follows:

1. Read at least the first 8 bytes that comprise the message header. The first 4 bytes are a message header identifier and the second 4 bytes are the length of the entire message in network byte order.
2. Using the `ntohl()` system function, convert the 4-byte message length into the host integer format.
3. Using the length, determine if the entire message has been received. If it has not, continue to receive data on the socket until the entire length of the message has arrived.
4. Pass the entire message buffer, including the header, as this parameter.

messageLen

The byte length of the alert message buffer supplied in the message parameter.

Return values

timestamp

The time that the alert event or indicator was created. Specify `NULL` for this parameter if you do not want to retrieve the time stamp.

sourceType

A value that identifies the source of the alert. The possible values are:.

`1:`	General purpose alert event (`AR_NOTIF_SOURCE_GP`).
`2:`	BMC Remedy Alert* event (`AR_NOTIF_SOURCE_AR`).
`3:`	First contact indicator (`AR_NOTIF_SOURCE_FIRST`).
`4:`	Accessibility check indicator (`AR_NOTIF_SOURCE_CHECK`).
`5:`	Flashboards alert event (`AR_NOTIF_SOURCE_FB`).

Specify `NULL` if you do not want to return the source type. * BMC Remedy Alert is no longer shipped with BMC Remedy AR System.

priority

A relative value that represents the priority for this alert. The range of acceptable values is between `0` and `10`. Specify `NULL` if you do not want to return the priority.

alertText

The text that the alert contains. Specify `NULL` if you do not want to return the text. If you do not specify `NULL`, you must free the returned memory buffer.

sourceTag

A string that identifies the source of the alert. Specify `NULL` if you do not want to return the source tag. If you do not specify `NULL`, you must free the returned memory buffer.

serverName

The name of the server that is the source of the alert. A value of `@` indicates the current server. Specify `NULL` if you do not want to return the server name. If you do not specify `NULL`, you must free the returned memory buffer.

serverAddr

The server IP address that the client expects. This is usually the IP address of the server named in the `serverName` parameter, except in the case where a load balancer works with the BMC Remedy AR System server or there are multiple servers accessing the same database. Specify `NULL` if you do not want to return the server address. If you do not specify `NULL`, you must free the returned memory buffer.

formName

The name of the form that is the source of the alert. For Flashboards, this is the name of the Flashboard that generated the alert. Specify `NULL` if you do not want to return this value. If you do not specify `NULL`, you must free the returned memory buffer.

objectId

The identifier for the object. For BMC Remedy AR System, this value is a special entry ID format. For a join form, the BMC Remedy AR System server concatenates all entry IDs and precedes them with a two-character count (for example, `0200000000000123450000000000023456`).

For non-join forms, the BMC Remedy AR System server precedes the entry IDs with a two-character count (`01000000000098765`). For Flashboards, this value is the name of the Flashboard alert. Specify `NULL` if you do not want to return this value. If you do not specify `NULL`, you must free the returned memory buffer.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent. See FreeAR for: `FreeARStatusList`.

6.2.20 ARDecodeARAssignStruct

Description

Converts a serialized assign string in a **DEF** file into an `ARAssignStruct` structure to facilitate string import.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDecodeARAssignStruct(
   ARControlStruct *control,
   char *assignText,
   ARAssignStruct *assignStruct,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is performed, and what session performs it. The **user** and **server** fields are required.

assignText

The serialized assign string to be converted.

Return values

assignStruct

The resulting `ARAssignStruct` structure.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDecodeARQualifierStruct, ARDecodeDiary, ARDecodeStatusHistory, AREncodeARAssignStruct, AREncodeARQualifierStruct, AREncodeDiary, AREncodeStatusHistory. See FreeAR for: `FreeARStatusList`.

6.2.21 ARDecodeARQualifierStruct

Description

Converts a serialized qualifier string into an `ARQualifierStruct` structure.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDecodeARQualifierStruct(
   ARControlStruct *control,
   char *qualText,
   ARQualifierStruct *qualStruct,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is performed, and what session performs it. The **user** and **server** fields are required.

qualText

The serialized qualification string to be converted.

Return values

qualStruct

The resulting `ARQualifierStruct` structure.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDecodeARAssignStruct, AREncodeARAssignStruct, AREncodeARQualifierStruct, ARDecodeDiary, AREncodeDiary, ARDecodeStatusHistory, AREncodeStatusHistory. See FreeAR for: `FreeARStatusList`.

6.2.22 ARDecodeDiary

Description

Parses any diary field (including the `changeDiary` associated with every BMC Remedy AR System object) into user, time stamp, and text components. The function takes the formatted string returned for all diary fields and decodes it into an array of diary entries.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDecodeDiary(
   ARControlStruct *control,
   char *diaryString,
   ARDiaryList *diaryList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

diaryString

A formatted diary string returned by `ARGetEntry` or any of the `ARGet` *Object* API functions.

Return values

diaryList

An array of decoded diary entries.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDecodeStatusHistory, ARGetActiveLink, ARGetCharMenu, ARGetEntry, ARGetField, ARGetFilter, ARGetSchema, ARGetVUI. See FreeAR for: `FreeARDiaryList`, `FreeARStatusList`, `AREncodeDiary`.

6.2.23 ARDecodeStatusHistory

Description

Parses the **Status History** core field into user and time stamp components. The function takes the formatted string returned for this field and decodes it into an array of status history entries.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDecodeStatusHistory(
   ARControlStruct *control,
   char *statHistString,
   ARStatusHistoryList *statHistList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

statHistString

A formatted status history string returned by `ARGetEntry`.

Return values

statHistList

An array of decoded status history entries.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDecodeDiary, ARGetEntry. See FreeAR for: `FreeARStatusList`, `FreeARStatusHistoryList`.

6.2.24 ARDeleteActiveLink

Description

Deletes the active link with the indicated name from the specified server and deletes any container references to the active link. The active link is deleted from the server immediately and is not returned to users who request information about active links. Because active links operate on clients, individual clients can continue using the active link until they reconnect to the form (thus reloading the form from the server).

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteActiveLink(
   ARControlStruct *control,
   ARNameType name,
   unsigned int deleteOption,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARDelete*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the active link to delete.

deleteOption

A bitmask that indicates the action to take.

Bit 0:	Server object default delete option. For locked objects, fails the delete attempt (`AR_DEFAULT_DELETE_OPTION`).
Bit 2:	Delete active link and all objects in the block that are locked with the same key. This is applicable only for locked objects (`AR_LOCK_BLOCK_DELETE`) .

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent, ARDeleteSchema, ARGetActiveLink, ARGetListActiveLink, ARGetMultipleActiveLinks, ARSetActiveLink. See FreeAR for: `FreeARStatusList`.

6.2.25 ARDeleteCharMenu

Description

Deletes the character menu with the indicated name from the specified server. The character menu is deleted from the server immediately and is not returned to users who request information about character menus. Because character menus operate on clients, individual clients can continue using the character menu until they reconnect to the form (thus reloading the form from the server).

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteCharMenu(
   ARControlStruct *control,
   ARNameType name,
   unsigned int deleteOption
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARDelete*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the character menu to delete.

deleteOption

A bitmask that indicates the action to take.

Bit 0:	Server object default delete option. For locked objects, fails the delete attempt (`AR_DEFAULT_DELETE_OPTION`).
Bit 2:	Delete character menu and all objects in the block (`AR_LOCK_BLOCK_DELETE`).

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateCharMenu, ARGetCharMenu, ARGetListCharMenu, ARSetCharMenu. See FreeAR for: `FreeARStatusList`, `ARExpandCharMenu`.

6.2.26 ARDeleteContainer

Description

Deletes the container with the indicated name from the specified server and deletes any references to the container from other containers. Objects referenced by the container are not deleted.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteContainer(
   ARControlStruct *control,
   ARNameType name,
   unsigned int deleteOption,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARDelete*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the container to delete.

deleteOption

A bitmask that indicates the action to take.

Bit 0:	Server object default delete option. For locked objects, fails the delete attempt (`AR_DEFAULT_DELETE_OPTION`).
Bit 2:	Delete container and all objects in the block (`AR_LOCK_BLOCK_DELETE`).

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

The `Rename` and `Delete` operations typically change multiple objects in addition to their primary target object. The `Rename` or `Delete` function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label

- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateContainer, ARGetContainer, ARGetListContainer, ARSetContainer. See FreeAR for: `FreeARStatusList`.

6.2.27 ARDeleteEntry

Description

Deletes the form entry with the indicated ID from the specified server. You can delete entries only from base forms. To remove entries from join forms, delete them from the underlying base forms.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteEntry(
   ARControlStruct *control,
   ARNameType schema,
   AREntryIdList *entryId,
   unsigned int option,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the base form containing the entry to delete.

entryId

A list of values of type `AREntryIdType` that identify the entries. For entries of join schemas, the list contains the entry IDs for all the member forms. For other schemas, the list contains one entry ID.

option

The policy that controls the deletion of entries for join schemas.

`0:`	Delete individual entries only when the entry can be retrieved through the join schema (`AR_JOIN_DELOPTION_NONE`).
`1:`	Delete individual entries even when the entry cannot be retrieved from the join schema (`AR_JOIN_DELOPTION_FORCE`). The BMC Remedy AR System server ignores errors for entries that no longer exist.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEntry, ARDeleteSchema, ARGetEntry, ARGetListEntry, ARSetEntry. See FreeAR for: `FreeAREntryIdList`, `FreeARStatusList`.

6.2.28 ARDeleteEscalation

Description

Deletes the escalation with the indicated name from the specified server and deletes any container references to the escalation. The escalation is deleted from the server immediately and is not returned to users who request information about escalations.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteEscalation(
   ARControlStruct *control,
   ARNameType name,
   unsigned int deleteOption,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARDelete*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the escalation to delete.

deleteOption

A bitmask that indicates the action to take.

Bit 0:	Server object default delete option. For locked objects, fails the delete attempt (`AR_DEFAULT_DELETE_OPTION`).
Bit 2:	Delete escalation and all objects in the block (`AR_LOCK_BLOCK_DELETE`).

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEscalation, ARDeleteSchema, ARGetEscalation, ARGetListEscalation, ARSetEscalation. See FreeAR for: `FreeARStatusList`.

6.2.29 ARDeleteField

Description

Deletes the form field with the indicated ID from the specified server. Depending on the value that you specify for the `deleteOption` parameter, the field is deleted from the server immediately and is not returned to users who request information about fields.

When a parent field is deleted, its child fields might also be deleted. For example, deleting a panel holder field deletes all panels within it. All fields within those panels are removed from the current view but not deleted. Deleting a table field deletes all columns within it.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteField(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalId fieldId,
   unsigned int deleteOption,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARDelete*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

schema

The name of the form containing the field to delete.

fieldId

The internal ID of the field to delete.

deleteOption

A value that indicates the action to take if the specified field contains data (applicable only to base forms) or is associated with a join form.

`0:`	Do not delete the field (`AR_FIELD_CLEAN_DELETE`).
`1:`	Delete if the field contains data but not if it is associated with a join form (`AR_FIELD_DATA_DELETE`).
`2:`	Delete the field, all join form fields that map to it, and all join forms dependent on it for join qualification (`AR_FIELD_FORCE_DELETE`).

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateField, ARDeleteMultipleFields, ARDeleteSchema, ARGetField, ARGetListField, ARGetMultipleFields, ARSetField. See FreeAR for: `FreeARStatusList`.

6.2.30 ARDeleteFilter

Description

Deletes the filter with the indicated name from the specified server and deletes any container references to the filter. The filter is deleted from the server immediately and is not returned to users who request information about filters.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteFilter(
   ARControlStruct *control,
   ARNameType name,
   unsigned int deleteOption,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the ARDelete* function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP variable of the ARSetSessionConfiguration function (see ARGetSessionConfiguration).

name

The name of the filter to delete.

deleteOption

A bitmask that indicates the action to take.

Bit 0:	Server object default delete option. For locked objects, fails the delete attempt (AR_DEFAULT_DELETE_OPTION).
Bit 2:	Delete filter and all objects in the block (AR_LOCK_BLOCK_DELETE) .

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- ARCreateSchema
- ARCreateMultipleFields
- ARSetVUI
- ARSetVUI
- ARSetSchema

In this case, the objectModificationLogLabel value should be passed only to the last call, ARSetSchema, even though the user provides the label during the ARCreateSchema operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateFilter, ARDeleteSchema, ARGetFilter, ARGetListFilter, ARSetFilter. See FreeAR for: `FreeARStatusList`.

6.2.31 ARDeleteImage

Description

Deletes the image with the indicated name from the specified server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteImage(
   ARControlStruct *control,
   ARNameType name,
   ARBoolean updateRef
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARDelete*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the image to delete.

updateRef

Specify `TRUE` to remove all references to the image. Specify `FALSE` to leave references to the image unchanged.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

The `Rename` and `Delete` operations typically change multiple objects in addition to their primary target object. The `Rename` or `Delete` function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateImage, ARGetImage, ARGetListImage, ARSetImage. See FreeAR for: `FreeARStatusList`.

6.2.32 ARDeleteLicense

Description

Deletes an entry from the Add or Remove Licenses form for the current server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteLicense(
   ARControlStruct *control,
   ARLicenseNameType licenseType,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

licenseType

The type of license to delete. See the **arstruct.h** file for license types.

licenseKey

The key associated with the license to delete. Required only for server licenses. (If your system has multiple server licenses, this key identifies which server license to delete.)

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateLicense, ARGetListLicense, ARValidateLicense, ARValidateMultipleLicenses. See FreeAR for: `FreeARStatusList`.

6.2.33 ARDeleteMultipleFields

Description

Deletes the form fields with the indicated IDs from the specified server. Depending on the value that you specify for the `deleteOption` parameter, the fields are deleted from the server immediately and are not returned to users who request information about fields. This function performs the same action as `ARDeleteField` but is easier to use and more efficient than deleting multiple entries one by one.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteMultipleFields(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalIdList *fieldList,
   unsigned int deleteOption,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARDelete*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

schema

The name of the form containing the fields to delete.

fieldList

The internal IDs of the fields to delete.

deleteOption

A value that indicates the action to take if one of the specified fields contains data (only applicable to base forms) or are associated with a join form.

`0:`	Do not delete the field (`AR_FIELD_CLEAN_DELETE`).
`1:`	Delete if the field contains data but not if it is associated with a join form (`AR_FIELD_DATA_DELETE`).
`2:`	Delete the field, all join form fields that map to it, and all join forms dependent on it for join qualification (`AR_FIELD_FORCE_DELETE`).

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateField, ARDeleteField, ARDeleteSchema, ARGetField, ARGetListField, ARGetMultipleFields, ARSetField. See FreeAR for: `FreeARInternalIdList`, `FreeARStatusList`.

6.2.34 ARDeleteSchema

Description

Deletes the form with the indicated name from the specified server and deletes any container references to the form. Depending on the value that you specify for the `deleteOption` parameter, the form is deleted from the server immediately and is not returned to users who request information about forms. In addition, the system deletes all entries, fields, views (VUIs), active links, escalations, and filters associated with the form and all containers owned by the form.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteSchema(
   ARControlStruct *control,
   ARNameType name,
   unsigned int deleteOption,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARDelete*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

name

The name of the form to delete.

deleteOption

A bitmask that indicates the action to take if the specified form contains entries (only applicable to base forms) or is a member of a join form.

Bit 0:	Delete schema only when there is no dependency and there is no data in the schema, set the bit to zero (0). This is ignored in the case of a join form (`AR_SCHEMA_CLEAN_DELETE`). To delete schema even if there is data, set the bit to one (1). This is applicable only to regular schemas (`AR_SCHEMA_DATA_DELETE`).

Bit 1:	Delete schema even if there is a dependency. All the specified schemas are deleted. This option overrides `AR_SCHEMA_DATA_DELETE` (`AR_SCHEMA_FORCE_DELETE`).
Bit 2:	Delete the given object and all objects that are locked with the same key. This is applicable only to locked objects (`AR_LOCK_BLOCK_DELETE`).
Bit 3:	Delete the archive form even if archive is enabled or the archive form if part of a lock block (`AR_SCHEMA_SHADOW_DELETE`).

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

The `Rename` and `Delete` operations typically change multiple objects in addition to their primary target object. The `Rename` or `Delete` function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateSchema, ARGetSchema, ARGetListAlertUser, ARSetSchema. See FreeAR for: `FreeARStatusList`.

6.2.35 ARDeleteSupportFile

Description

Deletes a support file in the BMC Remedy AR System.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteSupportFile(
   ARControlStruct *control,
   unsigned int fileType,
   ARNameType name,
   ARInternalId id2,
   ARInternalId fileId,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

fileType

The numerical value for the type of file, and the type of object the file is related to. Specify `1` (`AR_SUPPORT_FILE_EXTERNAL_REPORT`) for an external report file associated with an active link.

name

The name of the object the file is associated with, usually a form.

id2

The ID of the field or VUI, if the object is a form. If the object is not a form, set this parameter to `0`.

fileId

The unique identifier of a file within its object.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent, ARCreateSupportFile, ARDeleteActiveLink, ARGetActiveLink, ARGetListSupportFile, ARGetSupportFile, ARSetActiveLink, ARSetSupportFile. See FreeAR for: `FreeARStatusList`.

6.2.36 ARDeleteVUI

Description

Deletes the form view (VUI) with the indicated ID from the specified server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeleteVUI(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalId vuiId,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARDelete*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARGetSessionConfiguration).

schema

The name of the form containing the VUI to delete.

vuiId

The internal ID of the VUI to delete.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateVUI, ARDeleteSchema, ARGetVUI, ARGetListVUI, ARSetVUI. See FreeAR for: `FreeARStatusList`.

6.2.37 ARDeregisterForAlerts

Description

Cancels registration for the specified user on the specified BMC Remedy AR System server and port. The system deletes the user from the active list and stops delivering alerts.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARDeregisterForAlerts
```

```
    ARControlStruct *control,
    int clientPort,
    ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

clientPort

The TCP port that the client uses. Client programs can use different port numbers to register with the same server multiple times, but must deregister individually.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARRegisterForAlerts, ARCreateAlertEvent. See FreeAR for: `FreeARStatusList`.

6.2.38 AREncodeARAssignStruct

Description

Converts an `ARAssignStruct` structure into a serialized assignment string. The resulting string is stored in a **DEF** file and is used for exporting definitions of `ARServer` objects.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int (AREncodeARAssignStruct
   ARControlStruct *control,
   ARAssignStruct *assignStruct,
   char **assignText,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is performed, and what session performs it. The **user** and **server** fields are required.

assignStruct

The `ARAssignStruct` structure to convert.

Return values

assignText

A serialized representation of the assignment structure.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDecodeARQualifierStruct, AREncodeARAssignStruct, AREncodeARQualifierStruct, ARDecodeDiary, AREncodeDiary, ARDecodeStatusHistory, AREncodeStatusHistory. See FreeAR for: `FreeARStatusList`.

6.2.39 AREncodeARQualifierStruct

Description

Converts an `ARQualifierStruct` into a serialized qualification string. The resulting string is stored in a **DEF** file and is used for exporting definitions of `ARServer` objects.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
```

```
#include "arstruct.h"

int AREncodeARQualifierStruct(
   ARControlStruct *control,
   ARQualifierStruct *qualStruct,
   char **qualText,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

qualStruct

The `ARQualifierStruct` structure to convert.

Return values

qualText

A serialized representation of the qualification structure.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDecodeARQualifierStruct, AREncodeARAssignStruct, AREncodeARQualifierStruct, ARDecodeDiary, AREncodeDiary, ARDecodeStatusHistory, AREncodeStatusHistory. See FreeAR for: `FreeARStatusList`.

6.2.40 AREncodeDiary

Description

Converts an `ARDiaryList` into a diary string. The resulting string is stored in a **DEF** file and is used for exporting definitions of `ARServer` objects.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int AREncodeDiary(
   ARControlStruct *control,
   ARDiaryList *diaryList,
   char **diaryString,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

diaryList

The `ARDiaryList` to convert.

Return values

diaryString

The resulting diary string.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDecodeARQualifierStruct, AREncodeARAssignStruct, AREncodeARQualifierStruct, ARDecodeDiary, AREncodeDiary, ARDecodeStatusHistory, AREncodeStatusHistory. See FreeAR for: `FreeARStatusList`.

6.2.41 AREncodeStatusHistory

Description

Converts an `ARStatusHistoryList` into a status history string. The resulting string is stored in a **DEF** file and is used for exporting definitions of `ARServer` objects.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int AREncodeStatusHistory(
   ARControlStruct *control,
   ARStatusHistoryList *statHistList,
   char **statHistString,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

statHistList

The `ARStatusHistoryList` structure to convert.

Return values

statHistString

The resulting status history string.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDecodeARQualifierStruct, AREncodeARAssignStruct, AREncodeARQualifierStruct, ARDecodeDiary, AREncodeDiary, ARDecodeStatusHistory, AREncodeStatusHistory. See FreeAR for: `FreeARStatusList`.

6.2.42 AREndBulkEntryTransaction

Description

Marks the ending of a series of entry API function calls that are grouped together and sent to the BMC Remedy AR System server as part of one transaction. All calls related to create, set, delete, and merge operations made before the call to this function and after the preceding `ARBeginBulkEntryTransaction` call are sent to the server when this function is called and executed within a single database transaction.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int AREndBulkEntryTransaction(
   ARControlStruct *control,
   unsigned int actionType,
   ARBulkEntryReturnList *bulkEntryReturnList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

actionType

The action to take. Use `AR_BULK_ENTRY_ACTION_SEND` to transmit the queued entry calls to the server and end the transaction. Use `AR_BULK_ENTRY_ACTION_CANCEL` to remove the queued entry calls and end the transaction.

Return values

bulkEntryReturnList

A list of return structures corresponding to the individual operations executed within the bulk entry transaction. This parameter can be `NULL`. If it is not `NULL`, this parameter returns information regardless of whether the status parameter indicates an error occurred. The status of the individual calls is returned in the order the calls were issued, up to and including the last call executed on the server. If any of the individual calls fail, the list terminates with the return structure for the failed call. Any return information from previous calls that were successful can be ignored.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking. If any of the individual entry calls fail during the execution of the API call, the error code `9713` (`AR_ERROR_BULK_ENTRY_CALL_FAILED`) is returned and the transaction is rolled back. The status of each individual entry call is returned in the `bulkEntryReturnList`.

See also

ARBeginBulkEntryTransaction.

6.2.43 ARExecuteProcess

Description

Performs the indicated command on the specified server. Depending on the values that you specify for the `returnStatus` and `returnString` parameters, you can execute the command as an independent process or wait for the process to complete and return the result to the client. The system executes the command based on the access privileges of the user who launched the BMC Remedy AR System server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARExecuteProcess(
   ARControlStruct *control,
   char *command,
   int *returnStatus,
   char **returnString,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

command

The command to execute (must be a valid command on the specified server).

Return values

returnStatus

An integer that specifies the status of the operation. A value of `0` generally indicates success. Any other value generally indicates a failure. Specify `NULL` for this parameter and the `returnString` parameter if you want the system to launch an independent process and not wait for it to complete. Otherwise, specify a value for this parameter if you want the system to wait for the process to complete before returning. If the process does not finish within the time-out interval, adjust the filter process time-out interval to prevent server blocking. For information about the process time-out interval, see Setting timeout options.

returnString

A string containing the process output. Depending on the outcome of the operation, this string contains either result data or an error message. Specify `NULL` for this parameter and the `returnStatus` parameter if you want the system to launch an independent process and not wait for it to complete. Otherwise, specify a value for this parameter if you want the system to wait for the process to complete before returning. If the process does not finish within the time-out interval, adjust the filter process time-out interval to prevent server blocking (configurable from 1 to 20 seconds).

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListSQL. See FreeAR for: `FreeARStatusList`.

6.2.44 ARExpandCharMenu

Description

Expands the references for the specified menu definition and returns a character menu with only list-type items.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All query items, therefore, are limited to fields the user can access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"
```

```
int ARExpandCharMenu(
   ARControlStruct *control,
   ARCharMenuStruct *menuIn,
   unsigned int maxRetrieve,
   unsigned int *numMatches,
   ARCharMenuStruct *menuOut,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

menuIn

The menu definition to expand.

maxRetrieve

The maximum number of entries to retrieve for the Search menu. Use this parameter to limit the amount of data returned if the menu qualification does not sufficiently narrow the list. Specify `0` (`AR_NO_MAX_LIST_RETRIEVE`) to assign no maximum.

Return values

numMatches

The total number of (accessible) entries that match the menu qualification criteria. This value does not represent the number of entries returned unless the number of matching entries is less than or equal to the `maxRetrieve` value. Specify `NULL` for this parameter if you do not want to retrieve this value.

Performing this count requires additional search time if the number of matching entries is more than the `maxRetrieve` value.

menuOut

The expanded character menu. The menu definition contains only list-type items.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateCharMenu, ARDeleteCharMenu, ARGetCharMenu, ARGetListCharMenu, ARSetCharMenu. See FreeAR for: `FreeARCharMenuStruct`, `FreeARStatusList`.

6.2.45 ARExport

Description

Exports the indicated structure definitions from the specified server. Use this function to copy structure definitions from one BMC Remedy AR System server to another (see ARImport).

Note

Form exports do not work the same way with `ARExport` as they do in BMC Remedy Developer Studio. Other than views, you cannot automatically export related items along with a form. You must explicitly specify the workflow objects that you want to export. Also, `ARExport` cannot export a form without embedding the server name in the export file (something you can do with the "Server-Independent" option in BMC Remedy Developer Studio).

Privileges

For filters, escalations, character menus, and distributed mappings, this operation can be performed only by users with BMC Remedy AR System administrator privileges. For forms, containers, and active links, this operation can be performed by users with access permission for the specified structure. Access to `groupList` information for these structures is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARExport(
   ARControlStruct *control,
   ARStructItemList *structItems,
   ARNameType displayTag,
   unsigned int vuiType,
   unsigned int exportOption,
   ARWorkflowLockStruct *lockinfo,
   char **exportBuf,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

The runtime overlay group setting in the control record has the following effects:

- When the runtime group specifies a particular overlay group, exporting overlaid objects also exports their overlays, even if the overlays are not explicitly selected for export.
- When the runtime group is set to `-1`, only overlaid objects, not their overlays, are exported.
- When the runtime group is set to `-2`, unmodified, overlaid, overlay, and custom objects are exported using their real names.

Note

If inherited attributes are exported, they are ignored and their values are taken from the origin object on the server. Extended (additive) and overwritten attributes are always exported.

For information about runtime groups and the overlay group setting, see ARSetSessionConfiguration.

structItems

A list of zero or more structure items to export (identified by type and name). See Importing and exporting and structures.

displayTag

The label of the form view (VUI) to export. You must specify this parameter to export a particular view (`AR_STRUCT_ITEM_SCHEMA_VIEW`). If the specified view does not exist (or you specify `NULL` for this value), the system exports the default view. The system exports the first view in the list if the form does not have a default view.

vuiType

The type of VUI to export.

`0:`	No VUI type (`AR_VUI_TYPE_NONE`).
`1:`	Windows type (`AR_VUI_TYPE_WINDOWS`) ***** - fields in BMC Remedy User.
`2:`	Web relative type (`AR_VUI_TYPE_WEB`) - fields in view can be adjusted.
`3:`	Web absolute type (`AR_VUI_TYPE_WEB_ABS_POS`) - fields in view are fixed.

Fields marked with a ***** are available for legacy environments that use BMC Remedy User, which is no longer shipped with BMC Remedy AR System.

exportOption

A bitmask to manage export settings when exporting a deployable application. Use this parameter to specify shared workflow, integration workflow, and locale. This parameter takes the following values:

Bit	Symbol	Description
0	`EXPORT_DEFAULT`	Export all views, data and workflow for all objects in the application, but not integration workflow.
1	`AR_EXPORT_SHARED_WORKFLOW`	Export the default objects plus workflow attached to forms that are in the application but are not the primary form of the workflow (integration workflow).
2	`AR_EXPORT_DEFAULT_LOCALE`	For views and data, export only the default locale.
4	`AR_EXPORT_SELECTED_LOCALES`	Export locale specific views and data for each of the locales specified in the `selectedElements` list as specified in the `structItems` argument.
8	`AR_EXPORT_APP_INTEGRATION_WORKFLOW`	Export only the integration workflow between the exported application and the application specified in the `selectedElements` list as specified in the `structItems` argument.
16	`AR_EXPORT_LOCALE_ONLY`	Export only the locale-specific views and data, but not schemas or workflow.

`AR_EXPORT_DEFAULT_LOCALE` and `AR_EXPORT_SELECTED_LOCALES` control what locales are exported. `AR_EXPORT_LOCALE_ONLY` controls what server objects are exported.

lockInfo

To export objects as *locked*, you must supply this parameter. If exporting in unlocked mode, you can set this to `NULL`.

Note

The locking feature is for protecting your intellectual property. You can now export workflow as locked (either read-only lock or hidden-lock) and deliver it to your customers. When your customer imports the workflow, the system seamlessly runs that workflow but restrict the clients from getting the definition data, thus preventing human administrators or API clients from viewing or modifying the definition.

The values consist of a lock type and lock key. Valid values for the lock type are:

`0:`	Export the objects as unlocked (`AR_LOCK_TYPE_NONE`).
`1:`	Export the objects with read-only lock. When these objects are imported, they are readable but not modifiable (`AR_LOCK_TYPE_READONLY`).
`2:`	Export the objects with hidden-lock. When these objects are imported, they are neither readable nor modifiable (`AR_LOCK_TYPE_HIDDEN`).

Lock key is a string that is to be used as a key (or a password) to enforce locking.

Return values

exportBuf

A buffer that contains the definition text for the items specified for the `structItems` parameter. The system returns error messages for items not exported due to error.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARExportToFile, ARImport, See FreeAR for: `FreeARStatusList`, `FreeARStructItemList`.

6.2.46 ARExportLicense

Description

Specifies a pointer that is set to the memory allocated space and contains the full contents of the Add or Remove Licenses form currently on the server. This buffer can be written to a file to produce an exact replication of the Add or Remove Licenses form, including all checksums and encryption.

Privileges

BMC Remedy AR System administrator

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARExportLicense(
   ARControlStruct *control,
   char **exportBuf,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

exportBuf

The contents of the Add or Remove Licenses form formatted as the text of a license file.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARImportLicense.

6.2.47 ARExportToFile

Description

Exports the indicated structure definitions from the specified server to a file. Use this function to copy structure definitions from one BMC Remedy AR System server to another.

Note

Form exports do not work the same way with `ARExportToFile` as they do in BMC Remedy Developer Studio. Other than views, you cannot automatically export related items along with a form. You must explicitly specify the workflow items that you want to export. Also, `ARExportToFile` cannot export a form without embedding the server name in the export file (something you can do with the **Server-Independent** option in BMC Remedy Developer Studio).

Privileges

For filters, escalations, character menus, and distributed mappings, this operation can be performed only by users with BMC Remedy AR System administrator privileges. For forms, containers, and active links, this operation can be performed by users with access permission for the specified structure. Access to `groupList` information for these structures is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARExportToFile(
   ARControlStruct *control,
   ARStructItemList *structItems,
   ARNameType displayTag,
   unsigned int vuiType,
   ARWorkflowLockStruct *lockinfo,
   FILE *filePtr,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

structItems

A list of zero or more structure items to export (identified by type and name). See Importing and exporting and structures.

displayTag

The label of the form view (VUI) to export. You must specify this parameter to export a particular view (`AR_STRUCT_ITEM_SCHEMA_VIEW`). If the specified view does not exist (or you specify `NULL` for this value), the system exports the default view. The system exports the first view in the list if the form does not have a default view.

vuiType

The type of VUI to export.

`0:`	No VUI type (`AR_VUI_TYPE_NONE`).
`1:`	Windows type (`AR_VUI_TYPE_WINDOWS`) ***** - fields in BMC Remedy User.
`2:`	Web relative type (`AR_VUI_TYPE_WEB`) - fields in view can be adjusted.
`3:`	Web absolute type (`AR_VUI_TYPE_WEB_ABS_POS`) - fields in view are fixed.

Fields marked with an asterisk (*) are available for legacy environments that use BMC Remedy User, which is no longer shipped with BMC Remedy AR System.

lockInfo

To export objects as *locked*, you must supply this parameter. If exporting in unlocked mode, you can set this to `NULL`.

Note

The locking feature is for protecting your intellectual property. You can now export workflow as locked (either read-only lock or hidden-lock) and deliver it to your customers. When your customer imports the workflow, the system seamlessly runs that workflow but restrict the clients from getting the definition data, thus preventing human administrators or API clients from viewing or modifying the definition.

The values consist of a lock type and lock key. Valid values for the lock type are:

`0:`	Export the objects as unlocked (`AR_LOCK_TYPE_NONE`).
`1:`	Export the objects with read-only lock. When these objects are imported, they are readable but not modifiable (`AR_LOCK_TYPE_READONLY`).
`2:`	Export the objects with hidden-lock. When these objects are imported, they are neither readable nor modifiable (`AR_LOCK_TYPE_HIDDEN`).

Lock key is a string that is to be used as a key (or a password) to enforce the locking.

filePtr

A pointer to the export file to be created in the system.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

See ARExport, ARImport, FreeAR for: `FreeARStatusList`, `FreeARStructItemList`.

6.2.48 ARGetActiveLink

Description

Retrieves information about the active link with the indicated name from the specified server.

Privileges

This operation can be performed by users with access permission for the active link. Access to `groupList` information is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetActiveLink(
   ARControlStruct *control,
   ARNameType name,
   unsigned int *order,
   ARWorkflowConnectStruct *schemaList,
   ARInternalIdList *assignedGroupList,
   ARInternalIdList *groupList,
   unsigned int *executeMask,
   ARInternalId *controlField,
   ARInternalId *focusField,
   unsigned int *enable,
   ARQualifierStruct *query,
   ARActiveLinkActionList *actionList,
   ARActiveLinkActionList *elseList,
   char **helpText,
   ARTimestamp *timestamp,
   ARAccessNameType owner,
   ARAccessNameType lastChanged,
   char **changeDiary,
   ARPropList *objPropList,
   unsigned int *errorActlinkOptions,
   ARNameType errorActlinkName,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

name

The name of the active link to retrieve.

Return values

order

A value between `0` and `1000`(inclusive) that determines the active link execution order. When multiple active links are associated with a form, the value associated with each active link determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

schemaList

The list of form names the active link is linked to. The active link must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve the list.

assignedGroupList

A list of zero or more groups that are directly assigned permission to the active link.

groupList

A list of zero or more groups who can access this active link, including the inheritance hierarchy in the case of hierarchical groups. Access to this information is limited to users with BMC Remedy AR System administrator privileges. Specify `NULL` for this parameter if you do not want to retrieve this value.

If the static inheritance property is not enabled for the active link, this list will be the same as the `assignedGroupList`.

executeMask

A bitmask that indicates the form operations that trigger the active link. See ARCreateActiveLink for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this value.

controlField

The ID of the field that represents the button, toolbar button, or menu item associated with executing the active link. The system returns zero if the `executeMask` does not include the `AR_EXECUTE_ON_BUTTON` condition. Specify `NULL` for this parameter if you do not want to retrieve this value.

focusField

The ID of the field associated with executing the active link by pressing Return or selecting a character menu item. The system returns zero if the `executeMask` does not include the `AR_EXECUTE_ON_RETURN` or `AR_EXECUTE_ON_MENU_CHOICE` conditions. Specify `NULL` for this parameter if you do not want to retrieve this value.

enable

A flag that specifies whether the active link is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A qualification that determines whether the active link is executed. The system returns zero (`AR_COND_OP_NONE`) if the active link has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

actionList

The set of actions performed if the condition defined by the `query` parameter is satisfied. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseList

The set of actions performed if the condition defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

helpText

The help text associated with the active link. Specify `NULL` for this parameter if you do not want to retrieve the help text. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

timestamp

A time stamp that specifies the last change to the active link. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The active link owner. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChanged

The user who made the last change to the active link. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiary

The change diary associated with the active link. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties list is returned. See Server object properties.

errorActlinkOptions

Reserved for future use. Set to `NULL`.

errorActlinkName

Reserved for future use. Set to NULL.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent, ARDecodeDiary, ARDeleteActiveLink, ARGetListActiveLink, ARGetMultipleActiveLinks , ARSetActiveLink. See FreeAR for: FreeARActiveLinkActionList, FreeARInternalIdList, FreeARPropList, FreeARQualifierStruct, FreeARStatusList.

6.2.49 ARGetAlertCount

Description

Retrieves the count of qualifying alert events located from the specified server.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetAlertCount(
   ARControlStruct *control,
   ARQualifierStruct *qualifier,
   unsigned int *count,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

qualifier

The qualifying criteria for the alert events that this function counts. Specify NULL or assign an operation value of 0 (AR_COND_OP_NONE) to count all alert events.

Return values

count

The number of alert events that meet the criteria that the `qualifier` argument specifies.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARRegisterForAlerts, ARCreateAlertEvent. See FreeAR for: `FreeARStatusList`.

6.2.50 ARGetApplicationState

Description

Retrieves the application state: maintenance (admin only), test, or production.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetApplicationState(
   ARControlStruct *control,
   ARNameType applicationName,
   ARNameType currentStateName,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

applicationName

The name of the application.

Return values

currentStateName

The current value for the state field, **currentStateName**, on the BMC Remedy AR System Application State form.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetApplicationState, ARGetListApplicationState.

6.2.51 ARGetCacheEvent

Description

Retrieves the list of events that occurred in the BMC Remedy AR System server cache and the number of server caches, and indicates when administrative operations become public. You can direct this API call to either return the results immediately or when the next event occurs. This call is useful for detecting cache events in production cache mode.

A server cache event occurs when you perform an administrative operation (for example, importing a definition file, creating a schema, creating workflow, or creating other BMC Remedy AR System objects). When the operation is performed, BMC Remedy AR System creates a new copy of the server cache (administrative cache) and modifies it. After a brief delay, the old cache is released and the new cache is made public, so that newly created objects are available to users. If multiple operations are performed simultaneously, BMC Remedy AR System creates multiple copies of the administrative cache.

This call differs from other API calls in that it has a default 8 hour timeout.

Privileges

The system returns data based on the access privileges of the user that you specify for the `control` parameter.

Synopsis

```
#include "ar.h"

int ARGetCacheEvent(
   ARControlStruct *control,
   ARInternalIdList *eventIdList,
   unsigned int returnOption,
   ARInternalIdList *eventIdOccuredList,
   unsigned int *cacheCount,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

eventIdList

A list of types of events to return if they have occurred:

`1:`	Indicates an "admin-only cache created" event. (`AR_CACHE_ADMINONLYCREATE_EVENT`)
`2:`	Indicates an "admin-only cache made public" event. (`AR_CACHE_ADMINONLYPUBLIC_EVENT`)
`3:`	Indicates an "cache freed" event. (`AR_CACHE_FREE_EVENT`)
`4:`	Indicates a "Group re-cache" event. (`AR_CACHE_GROUPCHANGEPUBLIC_EVENT`)

returnOption

Controls whether the call returns immediately or waits for the next event to occur:

`0:`	Default; the call returns the information immediately. (`AR_GCE_OPTION_NONE`)
`1:`	The call waits for the next event to occur before returning the information. (`AR_GCE_OPTION_NEXT`)

In the case of `AR_GCE_OPTION_NONE`, the server immediately returns with the list of events that are currently true.

In the case of `AR_GCE_OPTION_NEXT`, the server only returns when one of the events in the list becomes true. At that time, the list of returned events is the one that caused the call to return.

Note

In developer cache mode, the call always returns immediately regardless of the value of the return option. In developer cache mode, there can only ever be one copy of the cache and it is always public.

Return values

eventIdOccuredList

The list of types of events that have occurred. The possible events are:

AR_CACHE_ADMINONLYCREATE_EVENT	An admin-only cache has been created.
AR_CACHE_ADMINONLYPUBLIC_EVENT	An admin-only cache has become public.
AR_CACHE_FREE_EVENT	A copy of the cache has been freed.

cacheCount

The number of server caches. The call always returns the current number of copies of the cache in memory at the time that it returns.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.52 ARGetCharMenu

Description

Retrieves information about the character menu with the indicated name from the specified server.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetCharMenu(
   ARControlStruct *control,
   ARNameType name,
   unsigned int *refreshCode,
   ARCharMenuStruct *menuDefn,
   char **helpText,
   ARTimestamp *timestamp,
   ARAccessNameType owner,
   ARAccessNameType lastChanged,
   char **changeDiary,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see description for Overlay-mode in ar.cfg or ar.conf options N-R.

name

The name of the character menu to retrieve.

Return values

refreshCode

A value that indicates when the menu is refreshed. See ARCreateCharMenu for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this value.

menuDefn

The definition of the character menu. Specify `NULL` for this parameter if you do not want to retrieve this value.

helpText

The help text associated with the character menu. Specify `NULL` for this parameter if you do not want to retrieve the help text. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

timestamp

A time stamp that specifies the last change to the character menu. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owning user for the character menu. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChanged

The user who made the last change to the character menu. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiary

The change diary associated with the character menu. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a properties list with zero properties is associated with the object, and when an `ARGetCharMenu` action is performed, zero properties are returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateCharMenu, ARDecodeDiary, ARDeleteCharMenu, ARExpandCharMenu, ARGetListCharMenu, ARGetMultipleCharMenus, ARSetCharMenu. See FreeAR for: `FreeARCharMenuStruct`, `FreeARPropList`, `FreeARStatusList`.

6.2.53 ARGetClientCharSet

Description

Retrieves a string that represents the name of the character set the client is using. The API assumes that all character data the client passes it is encoded in this character set, and returns all character data encoded in this character set.

Use the `ARControlStruct localeInfo.charSet` field to set the client character set as described under `ARInitialization()`.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARGetClientCharSet(
   ARControlStruct *control,
   char *charSet,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **server** field is required.

Return values

charSet

A string that names the character set the client is using. `charSet` is a buffer whose size is at least `AR_MAX_LANG_SIZE+1`.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetServerCharSet

6.2.54 ARGetContainer

Description

Retrieves the contents of the container with the indicated name from the specified server. It can return references of a single, specified type, of all types, or of an exclude reference type. The system returns information for accessible references and does nothing for references for which the user does not have access.

Privileges

This operation can be performed by users with access permission for the container. Access to `groupList` and `admingrpList` information is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"
```

```
int ARGetContainer(
   ARControlStruct *control,
   ARNameType name,
   ARReferenceTypeList *refTypes,
   ARPermissionList *assignedGroupList
   ARPermissionList *groupList,
   ARInternalIdList *admingrpList,
   ARContainerOwnerObjList *ownerObjList,
   char **label,
   char **description,
   unsigned int *type,
   ARReferenceList *references,
   char **helpText,
   ARAccessNameType owner,
   ARTimestamp *timestamp,
   ARAccessNameType lastChanged,
   char **changeDiary,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the AR_SESS_CONTROL_PROP_API_OVERLAYGROUP variable of the ARSetSessionConfiguration function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the AR_SESS_CONTROL_PROP_API_OVERLAYGROUP variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

name

The name of the container to retrieve.

refTypes

A list of the types of references (for example, forms and filters) to retrieve. You can specify individual types of references to retrieve, specify that all (ARREF_ALL) or none (ARREF_NONE) of the references be retrieved, or specify negative numbers to treat the types of references as exclude reference types. The exclude reference types take precedence over the include list; this means that if you specify a type as positive as well as negative, then all references of that type are excluded.

Return values

assignedGroupList

A list of zero or more groups that are directly assigned permission to the container.

groupList

A list of zero or more groups who can access this container, including the inheritance hierarchy in the case of hierarchical groups. Access to this information is limited to users with BMC Remedy AR System administrator privileges. Specify `NULL` for this parameter if you do not want to retrieve this value.

If the static inheritance property is not enabled for the container, this list will be the same as the `assignedGroupList`.

admingrpList

A list of zero or more groups who can administer this container (and the referenced objects). If `ownerObj` does not return `NULL`, this list is the Subadministrator group list of the owning form. Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify `NULL` for this parameter if you do not want to retrieve this value.

ownerObjList

A list of the schemas that own this container. Specify `NULL` for this parameter if you do not want to retrieve this value. If this parameter returns `NULL`, the container exists globally.

label

The label for this container. Specify `NULL` for this parameter if you do not want to retrieve this value.

description

The description for this container. Specify `NULL` for this parameter if you do not want to retrieve this value.

type

The type for this container. Specify `NULL` for this parameter if you do not want to retrieve this value.

references

Pointers to the objects (for example, forms or filters) referenced by this container. Specify `NULL` for this parameter if you do not want to retrieve this value.

helpText

The help text associated with the container. Specify `NULL` for this parameter if you do not want to retrieve the help text. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

owner

The owning user for the container. Specify `NULL` for this parameter if you do not want to retrieve this value.

timestamp

A time stamp that specifies the last change to the container. Specify NULL for this parameter if you do not want to retrieve this value.

lastChanged

The user who made the last change to the container. Specify NULL for this parameter if you do not want to retrieve this value.

changeDiary

The change diary associated with the container. The server adds the user making the change and a time stamp when it saves the change diary. Use ARDecodeDiary to parse the change diary into user, time stamp, and text components. Specify NULL for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to NULL, a properties list with zero properties is associated with the object. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateContainer, ARDecodeDiary, ARDeleteContainer, ARGetListContainer, ARGetSchema, ARSetContainer. See FreeAR for: FreeARContainerInfoList, FreeARInternalIdList, FreeARPermissionList, FreeARPropList, FreeARReferenceList, FreeARStatusList.

6.2.55 ARGetCurrencyRatio

Description

Retrieves a selected currency ratio from a set of ratios returned when a program makes a call to ARGetMultipleCurrencyRatioSets.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
```

```
int ARGetCurrencyRatio(
   ARControlStruct *control,
   char *currencyRatios,
   ARCurrencyCodeType fromCurrencyCode
   ARCurrencyCodeType toCurrencyCode
   ARValueStruct *currencyRatio,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **sessionId** fields are required.

currencyRatios

A formatted currency ratio character string returned by a call to the `ARGetMultipleCurrencyRatioSets` function.

fromCurrencyCode

The source currency code for the conversion ratio.

toCurrencyCode

The target currency code for the conversion ratio.

Return values

currencyRatio

The conversion ratio for the specified source and target currency codes. The function returns a value structure of type `AR_DATA_TYPE_NULL` if there is no conversion ratio for the specified currency code combination. The function returns a value structure of type `AR_DATA_TYPE_DECIMAL` if there is a conversion ratio. You must free the `AR_DATA_TYPE_DECIMAL` value when it is no longer needed.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetMultipleCurrencyRatioSets. See FreeAR for: `FreeARValueStruct`.

6.2.56 ARGetEntry

Description

Retrieves the form entry with the indicated ID from the specified server. You can retrieve data for specific fields, all (accessible) fields, or no fields (which is useful to verify whether a form has any entries). This function returns only the name, size, and compressed size of attachment fields. Use `ARGetEntryBLOB` to retrieve the contents of the attachment.

Privileges

The system returns data based on the access privileges of the user that you specify for the `control` parameter. User permissions are verified for each specified field. The system returns values for accessible fields and warning messages for fields the user cannot access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetEntry(
   ARControlStruct *control,
   ARNameType schema,
   AREntryIdList *entryId,
   ARInternalIdList *idList,
   ARFieldValueList *fieldList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form containing the entry to retrieve.

entryId

The ID of the entry to retrieve.

Note

The system identifies entries in join forms by concatenating the entry IDs from the member forms. As a result, an entry ID can consist of one or more values of type `AREntryIdType` and, therefore, is represented by the `AREntryIdList` structure.

idList

A list of zero or more internal IDs that specify the fields to retrieve. Specify `NULL` for this parameter (or zero fields) to retrieve all (accessible) fields. Specify `NULL` for both this parameter and the `fieldList` parameter if you do not want to retrieve any fields. To minimize network traffic, specify only the fields that you need if you do not require the data for all fields. If an attachment field is specified in the list, only its name, size, and compressed size are returned. Use `ARGetEntryBLOB` to retrieve the contents of the attachment.

Return values

fieldList

A list of zero or more field/value pairs that identifies the data for the specified entry. The fields are returned in the order specified by `idList`. If the user does not have permission for a specified field or the field does not exist, the system does not return a value for the field/value pair. Specify `NULL` for this parameter if you do not want to retrieve any field data.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEntry, ARDecodeDiary, ARDecodeStatusHistory, ARDeleteEntry, ARGetEntryBLOB, ARGetListEntry, ARSetEntry. See FreeAR for: `FreeAREntryIdList`, `FreeARInternalIdList`, `FreeARFieldValueList`, `FreeARStatusList`.

6.2.57 ARGetEntryBLOB

Description

Retrieves the attachment, or binary large object (BLOB), stored for the attachment field with the indicated ID from the specified server. The BLOB can be placed in a buffer or a file.

Privileges

The system returns data based on the access privileges of the user that you specify for the `control` parameter. User permissions are verified for the specified field. If the user cannot access the field, the system returns an error message.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"
```

```
int ARGetEntryBLOB(
   ARControlStruct *control,
   ARNameType schema,
   AREntryIdList *entryId,
   ARInternalId id,
   ARLocStruct *loc,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form containing the entry to retrieve.

entryId

The ID of the entry to retrieve.

Note

The system identifies entries in join forms by concatenating the entry IDs from the member forms. As a result, an entry ID can consist of one or more values of type `AREntryIdType` and, therefore, is represented by the `AREntryIdList` structure.

id

The ID that specifies the field to retrieve.

loc

A pointer to an `ARLocStruct` structure that specifies how you want the contents of the blob returned: in a file (`AR_LOC_FILENAME`) or a data buffer (`AR_LOC_BUFFER`). The structure also contains the name of the file or buffer to be used.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetEntry, ARCreateEntry, ARDecodeDiary, ARDecodeStatusHistory, ARDeleteEntry, ARGetListEntry, ARSetEntry. See FreeAR for: `FreeAREntryIdList`, `FreeARInternalIdList`, `FreeARFieldValueList`, `FreeARStatusList`.

6.2.58 ARGetEntryBlock

Description

Retrieves a list of entries contained in a block of entries retrieved using `ARGetListEntryBlocks`.

Privileges

The system returns data based on the access privileges of the user that you specify for the `control` parameter. User permissions are verified for each specified field. The system returns values for accessible fields and warning messages for fields the user cannot access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetEntryBlock(
AREntryBlockList *entryBlockList,
   unsigned int blockNumber,
   AREntryListFieldValueList *entryList,
   ARStatusList *status)
```

Input arguments

entryBlockList

A list of entry blocks retrieved by `ARGetListEntryBlocks`.

blockNumber

The number of the block of entries for which you want to retrieve a list. A value of `0` represents the first block.

Return values

entryList

A list of entries contained in the specified block.

status

A list of zero or more notes, warnings, or errors generated from a call to this function.

See also

ARGetListEntryBlocks.

6.2.59 ARGetEntryStatistics

Description

Computes the indicated statistic for the form entries that match the conditions specified by the `qualifier` parameter and returns the values sorted in ascending order by the `groupByList.`

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All statistics, therefore, are limited to entries the user can access (users must have permission for the `entryId` field to access and retrieve entries).

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetEntryStatistics(
   ARControlStruct *control,
   ARNameType schema,
   ARQualifierStruct *qualifier,
   ARFieldValueOrArithStruct *target,
   unsigned int statistic,
   ARInternalIdList *groupByList,
   ARStatisticsResultList *results,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form to compute entry statistics for.

qualifier

A query that determines the set of entries to use. The qualification can include one or more fields and any combination of conditional, relational, and arithmetic (numeric data types only) operations. The system generates an error if the user does not have read permission for a field or a field does not exist. Specify `NULL` or assign an operation value of `0` (`AR_COND_OP_NONE`) to match all form entries.

target

The arithmetic operation that defines the statistic to compute. The statistic can include one or more fields and any combination of arithmetic operations. The system generates an error if the user does not have read permission for a field or a field does not exist. If you specify `AR_STAT_OP_COUNT` for the `statistic` parameter, assign a tag value of `0` to omit this parameter.

statistic

A value that indicates the statistic type.

`1:`	The total number of matching entries (`AR_STAT_OP_COUNT`).
`2:`	The sum of values for each group (`AR_STAT_OP_SUM`).
`3:`	The average value for each group (`AR_STAT_OP_AVERAGE`).
`4:`	The minimum value for each group (`AR_STAT_OP_MINIMUM`).
`5:`	The maximum value for each group (`AR_STAT_OP_MAXIMUM`).

groupByList

A list of zero or more fields to group the results by. The system computes a result for each group of entries having the same value in the specified field. Specifying more than one field creates groups within groups, each of which returns a separate statistic. Specify `NULL` for this parameter (or zero fields) to compute a single result for all matching entries.

Return values

results

A list of zero or more results sorted in ascending order. If you specify one or more fields for the `groupByList` parameter, each item in the list represents a group. Each result structure contains the field values that define the group and the statistic for that group.

For example, if the query was to find a count of bugs grouped by submitter and then by status, the results would return data sorted by submitter and then sorted by status, as in the following list:

```
Abe Low 1
Abe Medium 3
Dani Low 11
Dani Medium 10
Dani High 5
Sarah Low 8
Sarah High 2
```

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEntry, ARDeleteEntry, ARGetEntry, ARGetListEntry, ARLoadARQualifierStruct, ARMergeEntry, ARSetEntry. See FreeAR for: `FreeARFieldValueOrArithStruct`, `FreeARInternalIdList`, `FreeARQualifierStruct`, `FreeARStatisticsResultList`, `FreeARStatusList`.

6.2.60 ARGetEscalation

Description

Retrieves information about the escalation with the indicated name from the specified server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetEscalation(
   ARControlStruct *control,
   ARNameType name,
   AREscalationTmStruct *escalationTm,
   ARWorkflowConnectStruct *schemaList,
   unsigned int *enable,
   ARQualifierStruct *query,
   ARFilterActionList *actionList,
   ARFilterActionList *elseList,
   char **helpText,
   ARTimestamp *timestamp,
   ARAccessNameType owner,
   ARAccessNameType lastChanged,
   char **changeDiary,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

name

The name of the escalation to retrieve.

Return values

escalationTm

The time specification for evaluating the escalation condition. This parameter can take one of two forms: a time interval that defines how frequently the server checks the escalation condition (in seconds) or a bitmask that defines a particular day (by month or week) and time (hour and minute) for the server to check the condition. Specify `NULL` for this parameter if you do not want to retrieve this value.

schemaList

The list of form names the escalation is linked to. The escalation must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve the list.

enable

A flag that specifies whether the escalation is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A query operation performed when the escalation is executed that determines the set of entries to which the escalation actions (defined by the `actionList` parameter) are applied. The system returns `0` (`AR_COND_OP_NONE`) if the escalation has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

actionList

The set of actions performed for each entry that matches the criteria defined by the `query` parameter. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseList

The set of actions performed if no entries match the criteria defined by the `query` parameter. This list can contain from zero to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

helpText

The help text associated with the escalation. Specify `NULL` for this parameter if you do not want to retrieve the help text. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

timestamp

A time stamp that specifies the last change to the escalation. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owning user for the escalation. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChanged

The user who made the last change to the escalation. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiary

The change diary associated with the escalation. The server adds the user making the change and a time stamp when it saves the change diary. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. Specify `NULL` for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties list is returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEscalation, ARDecodeDiary, ARDeleteEscalation, ARGetListEscalation, ARGetMultipleEscalations, ARSetEscalation. See FreeAR for: `FreeARFilterActionList`, `FreeARPropList`, `FreeARQualifierStruct`, `FreeARStatusList`.

6.2.61 ARGetField

Description

Retrieves information about the form field with the indicated ID from the specified server.

Privileges

This operation can be performed by users with access permission for the field's parent form. Access to `permissions` information is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetField(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalId fieldId,
   ARNameType fieldName,
   ARFieldMappingStruct *fieldMap,
   unsigned int *dataType,
   unsigned int *option,
   unsigned int *createMode,
   unsigned int *fieldOption,
   ARValueStruct *defaultVal,
   ARPermissionList *assignedGroupList,
   ARPermissionList *permissions,
   ARFieldLimitStruct *limit,
   ARDisplayInstanceList *dInstanceList,
   char **helpText,
   ARTimestamp *timestamp,
   ARAccessNameType owner,
   ARAccessNameType lastChanged,
   char **changeDiary,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

schema

The name of the form containing the field to retrieve.

fieldId

The internal ID of the field to retrieve.

Return values

fieldName

he name of the field. Specify `NULL` for this parameter if you do not want to retrieve the field name.

fieldMap

A mapping to the underlying form from which to retrieve the field. Useful for join forms. Specify `NULL` for this parameter if you do not want to retrieve the field mapping. See Mapping fields in schemas.

dataType

The data type of the field. See ARCreateField for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve the data type.

option

A flag that indicates whether users must enter a value in the field. See ARCreateField for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this flag.

createMode

A flag that indicates the permission status for the field when users submit entries. See ARCreateField for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this flag.

fieldOption

A bitmask that indicates whether the field is to be audited or copied when other fields are audited.

Bit 0:	Audit this field. (`AR_FIELD_BITOPTION_AUDIT`)
Bit 1:	Copy this field when other fields in the form are audited. (`AR_FIELD_BITOPTION_COPY`)
Bit 2:	Indicates this field is for Log Key 1. (`AR_FIELD_BITOPTION_LOG_KEY1`)
Bit 3:	Indicates this field is for Log Key 2. (`AR_FIELD_BITOPTION_LOG_KEY2`)

Bits 2 and 3:	Indicates this field is for Log Key 3. (`AR_FIELD_BITOPTION_LOG_KEY3`)

defaultVal

The value to apply if a user submits an entry with no field value (only applicable to data fields). The system returns `0` (`AR_DEFAULT_VALUE_NONE`) if the field has no default. Specify `NULL` for this parameter if you do not want to retrieve the default value.

assignedGroupList

A list of zero or more groups that are directly assigned permission to the field.

permissions

A list of zero or more groups who can access this field, including the inheritance hierarchy in the case of hierarchical groups. Access to this information is limited to users with BMC Remedy AR System administrator privileges. Specify `NULL` for this parameter if you do not want to retrieve the permissions.

If the static inheritance property is not enabled for the field, this list will be the same as the `assignedGroupList`.

limit

The value limits for the field and other properties specific to the field's type. See Defining field limits for a description of the contained structure that applies to the type of field that you are creating. Specify `NULL` (or `AR_FIELD_LIMIT_NONE`) for trim and control fields or if you do not want to retrieve the limits and properties.

dInstanceList

A list of zero or more display properties associated with the field. See ARCreateField for a description of the possible values. The system returns `0` (`AR_DPROP_NONE`) if the field has no display properties. Specify `NULL` for this parameter if you do not want to retrieve this list.

helpText

The help text associated with the field. Specify `NULL` for this parameter if you do not want to retrieve the help text. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

timestamp

A time stamp that specifies the last change to the field. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owning user for the field. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChanged

The user who made the last change to the field. Specify NULL for this parameter if you do not want to retrieve this value.

changeDiary

The change diary associated with the field. The server adds the user making the change and a time stamp when it saves the change diary. Use ARDecodeDiary to parse the change diary into user, time stamp, and text components. Specify NULL for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to NULL, no object properties list is returned. See Server object properties and structures.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateField, ARDecodeDiary, ARDeleteField, ARGetMultipleFields, ARGetSchema, ARGetListField, ARSetField. See FreeAR for: FreeARDisplayInstanceList, FreeARFieldLimitList, FreeARPermissionList, FreeARStatusList, FreeARValueStruct.

6.2.62 ARGetFilter

Description

Retrieves information about the specified filter from the specified server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetFilter(
   ARControlStruct *control,
   ARNameType name,
   unsigned int *order,
   ARWorkflowConnectStruct *schemaList,
   unsigned int *opSet,
   unsigned int *enable,
   ARQualifierStruct *query,
   ARFilterActionList *actionList,
   ARFilterActionList *elseList,
   char **helpText,
   ARTimestamp *timestamp,
   ARAccessNameType owner,
   ARAccessNameType lastChanged,
   char **changeDiary,
   ARPropList *objPropList,
   unsigned int *errorFilterOptions
   ARNameType errorFilterName
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

name

The name of the filter to retrieve.

Return values

order

A value between `0` and `1000` (inclusive) that determines the filter execution order. When multiple filters are associated with a form, the value associated with each filter determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

schemaList

The list of form names the filter is linked to. The filter must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve the list.

opSet

A bitmask that indicates the form operations that trigger the filter. See ARCreateFilter for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve the operation set.

enable

A flag that specifies whether the filter is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this flag.

query

A qualification that determines whether the filter is executed. The system returns `0` (`AR_COND_OP_NONE`) if the filter has no qualification. Specify `NULL` for this parameter if you do not want to retrieve the query.

actionList

The set of actions performed if the condition defined by the `query` parameter is satisfied. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve the action list.

elseList

The set of actions performed if the condition defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve the else list.

helpText

The help text associated with the filter. Specify `NULL` for this parameter if you do not want to retrieve the help text. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

timestamp

A time stamp that specifies the last change to the filter. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owning user for the filter. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChanged

The user who made the last change to the filter. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiary

The change diary associated with the filter. The server adds the user making the change and a time stamp when it saves the change diary. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. Specify `NULL` for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties list is returned. See Server object properties.

errorFilterOptions

The error handler options for the filter. Returns `AR_FILTER_ERRHANDLER_ENABLE` if an error handler filter is enabled, zero if not. Specify `NULL` for this parameter if you do not want to retrieve this value.

errorFilterName

The name of error handler filter for this filter; `NULL` if none. Specify `NULL` for this parameter if you do not want to retrieve this value.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateFilter, ARDecodeDiary, ARDeleteFilter, ARGetListFilter, ARGetMultipleFilters, ARSetFilter. See FreeAR for: `FreeARFilterActionList`, `FreeARPropList`, `FreeARQualifierStruct`, `FreeARStatusList`.

6.2.63 ARGetImage

Description

Retrieves information about the specified image from the specified server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetImage(
   ARControlStruct *control,
   ARNameType name,
*ARImageDataStruct*
*content,
   char *imageType,
   ARTimeStamp *timestamp,
   char **checkSum,
   char **description,
   char **helpText,
   ARAccessNameType owner,
   char **changeDiary,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

name

The name of the image to retrieve.

Return values

content

The data for the image. See Images and structures. Specify `NULL` for this parameter if you do not want to retrieve this value.

imageType

The image encoding type stored as a character string. Valid values are: `BMP`, `GIF`, `JPEG` or `JPG`, and `PNG`. Specify `NULL` for this parameter if you do not want to retrieve this value.

timestamp

A time stamp that specifies the last change to the image. Specify `NULL` for this parameter if you do not want to retrieve this value.

checkSum

The MD5 hash value retrieved from the database for the image data in `content`. Your program can compute an MD5 has value and check it against this value to check whether the image data was retrieved correctly.

description

The description for this image. It can be as many as 2000 characters long or `NULL`. Specify `NULL` for this parameter if you do not want to retrieve this value.

helpText

The help text associated with the image. This text can be of any length. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owner for the image. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiary

The change diary associated with the image. The server adds the user making the change and a time stamp when it saves the change diary. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. Specify `NULL` for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties are returned. See Server object properties and structures.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateImage, ARDeleteImage, ARGetListImage, ARGetMultipleImages, ARSetImage. See FreeAR for: `FreeARImageDataStruct`, `FreeARStatusList`.

6.2.64 ARGetListActiveLink

Description

Retrieves a list of active link names from the specified server. You can retrieve all (accessible) active links or limit the list to active links associated with a particular form or modified after a specified time.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to active links the user can access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListActiveLink(
   ARControlStruct *control,
   ARNameType schema,
   ARTimestamp changedSince,
   ARPropList *objPropList,
   ARNameList *nameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

schema

The name of the form to retrieve active links for. Specify `NULL` for this parameter to retrieve active links for all forms.

changedSince

A time stamp that limits the active links retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve active links with any modification time stamp.

objPropList

List of server-managed object properties to search for. Returns all active links that match the object properties, for example, the application owner. See Server-managed object property tags.

Return values

nameList

A list of zero or more (accessible) active links that match the criteria in the `changedSince` and `objPropList` arguments. The system returns a list with zero items if no active links match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent, ARDeleteActiveLink, ARDeleteSchema, ARGetActiveLink, ARGetMultipleActiveLinks, ARSetActiveLink. See FreeAR for: `FreeARNameList`, `FreeARStatusList`.

6.2.65 ARGetListAlertUser

Description

Retrieves a list of all users that are registered for alerts on the specified server.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARGetListAlertUser(
   ARControlStruct *control,
   ARAccessNameList *userList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where the operation is performed, and which session the operation is performed in. The **user** and **server** fields are required.

Return values

userList

A list of zero or more user names registered to receive alerts on the specified server. The list returns only one entry for users that register under multiple addresses.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARRegisterForAlerts. See FreeAR for: `FreeARAccessNameList`, `FreeARStatusList`.

6.2.66 ARGetListApplicationState

Description

Retrieves the list of application states (maintenance, test, or production) that an application on this server can assume. This list is server-dependent.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListApplicationState(
   ARControlStruct *control,
   ARNameList *stateNameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

stateNameList

The list of states an application on this server can assume. There is one entry in the list for every field name on the BMC Remedy AR System Role Mapping from where the field ID resides in the Application State reserved range (`2000-2999`).

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetApplicationState, ARGetApplicationState.

6.2.67 ARGetListCharMenu

Description

Retrieves a list of character menu names from the specified server. You can retrieve all character menus or limit the list to character menus modified after a specified time.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListCharMenu(
   ARControlStruct *control,
   ARTimestamp changedSince,
   ARNameList *formList,
   ARNameList *actLinkList,
   ARPropList *objPropList
   ARNameList *nameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the character menus retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve character menus with any modification time stamp.

formList

A form name list that limits the character menus retrieved to the ones that are referenced by fields in the form.

actLinkList

An active link list that limits the character menus retrieved to the ones that have a change field action with the character menus.

objPropList

List of server-managed object properties to search for. Returns all character menus that match the object properties, for example, the application owner. See Server-managed object property tags.

Return values

nameList

A list of zero or more character menus that match the criteria defined in the changedSince, `formList`, `actLinkList`, and `objPropList` parameters. The system returns a list with zero items if no character menus match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateCharMenu, ARDeleteCharMenu, ARExpandCharMenu, ARGetCharMenu, ARSetCharMenu. See FreeAR for: `FreeARNameList`, `FreeARStatusList`.

6.2.68 ARGetListContainer

Description

Retrieves a list of containers from the specified server. You can retrieve all (accessible) containers or limit the list to containers of a particular type, containers owned by a specified form, or containers modified after a specified time.

Privileges

The system returns information based on the access privileges of the user that you specify for the **control** parameter. All lists, therefore, are limited to containers the user can access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListContainer(
   ARControlStruct *control,
   ARTimestamp changedSince,
   ARContainerTypeList *containerTypes,
   unsigned int attributes,
   ARContainerOwnerObjList *ownerObjList,
   ARPropList *objPropList,
   ARContainerInfoList *conList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the ARSetSessionConfiguration function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the containers retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve containers with any modification time stamp.

containerTypes

A list of values that indicates the container types to retrieve.

0:	Retrieve all container types (`ARCON_ALL`).
1:	Retrieve all guide containers (`ARCON_GUIDE`).
2:	Retrieve all application containers (`ARCON_APP`).
3:	Retrieve all packing list containers (`ARCON_PACK`).
4:	Retrieve all filter guide containers (`ARCON_FILTER_GUIDE`).

attributes

Specify `AR_HIDDEN_INCREMENT` for this parameter to retrieve both visible and hidden containers. Specify `NULL` for this parameter to retrieve only visible containers.

ownerObjList

A list of the structures that limit the containers retrieved, based on the object that owns them. Specify `NULL` for this parameter to reference all containers.

0:	Retrieve all globally owned containers (`ARCONOWNER_NONE`).
1:	Retrieve all containers (`ARCONOWNER_ALL`).
2:	Retrieve all containers owned by the specified form (`ARCONOWNER_SCHEMA`).

objPropList

List of server-managed object properties to search for. Returns all containers that match the object properties, for example, the application owner. See Server-managed object property tags.

Return values

conList

A list of zero or more (accessible) containers that match the criteria defined by the `containerTypes`, `ownerObj`, `changedSince`, and `objPropList` parameters. The system returns a list with zero items if no containers match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateContainer, ARDeleteContainer, ARGetContainer, ARGetListAlertUser, ARSetContainer. See FreeAR for: `FreeARNameList`, `FreeARStatusList`.

6.2.69 ARGetListEntry

Description

Retrieves a list of form entries from the specified server. The BMC Remedy AR System Server returns data from each entry as a string containing the concatenated values of selected fields. In the returned data, the combined length of all specified fields, including separator characters, can be up to 128 bytes (limited by `AR_MAX_SDESC_SIZE`). You can limit the list to entries that match particular conditions by specifying the `qualifier` parameter. `ARGetListEntryWithFields` can be used to return a qualified list of entries formatted as field/value pairs, without the 128 byte limitation.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to entries the user can access (users must have permission for the `entryId` field to access and retrieve entries).

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListEntry(
   ARControlStruct *control,
   ARNameType schema,
   ARQualifierStruct *qualifier,
   AREntryListFieldList *getListFields,
   ARSortList *sortList,
   unsigned int firstRetrieve,
   unsigned int maxRetrieve,
   ARBoolean useLocale,
   AREntryListList *entryList,
   unsigned int *numMatches,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form to retrieve entries for.

qualifier

A query that determines the set of entries to retrieve. The qualification can include one or more fields and any combination of conditional, relational, and arithmetic (numeric data types only) operations. The system generates an error if the user does not have read permission for a field or a field does not exist. Specify `NULL` or assign an operation value of `0` (`AR_COND_OP_NONE`) to match all form entries.

getListFields

A list of zero or more fields that identifies the data to display in the query list. The list can include any data fields except diary fields and long character fields. The system checks the permissions for each specified field and returns only those fields for which you have read access. The combined length of all specified fields, including separator characters, can be as many as 128 bytes (limited by `AR_MAX_SDESC_SIZE`). Specify `NULL` for this parameter (or zero fields) to return the default query list data for the form (see ARCreateSchema). The system returns the Short-Description core field if the form has no default query list data.

sortList

A list of zero or more fields that identifies the entry sort order. The system generates an error if you do not have read access on all specified fields. Specify `NULL` for this parameter (or zero fields) to use the default sort order for the form (see ARCreateSchema). The system sorts the entries in ascending order by `entryId` if the form has no default sort order.

firstRetrieve

The first entry to retrieve. A value of `0` (`AR_START_WITH_FIRST_ENTRY`) represents the first entry. A value of 1 will skip the first entry.

maxRetrieve

The maximum number of entries to retrieve. Use this parameter to limit the amount of data returned if the qualification does not sufficiently narrow the list. Specify `0` (`AR_NO_MAX_LIST_RETRIEVE`) to assign no maximum.

useLocale

A flag that indicates whether to search for entries based on the locale. If you specify `1` (`TRUE`) and the Localize Server option is selected, entries are searched using the locale specified in `AR_RESERV_LOCALE_LOCALIZED_SCHEMA`. If no matches are found for the specified locale, the search becomes less restrictive until a match is found. If you specify `0` (`FALSE`) or the Localize Server option is cleared, all entries are searched. For more information, see Setting the Localize Server option.

Return values

entryList

A list of zero or more (accessible) entries that match the criteria defined by the `qualifier` parameter. The system returns a list with zero items if no entries match the specified criteria.

numMatches

The total number of (accessible) entries that match the qualification criteria. This value does not represent the number of entries returned unless the number of matching entries is less than or equal to the `maxRetrieve` value. Specify `NULL` for this parameter if you do not want to retrieve this value.

Note

Performing this count requires additional search time if the number of matching entries is more than the `maxRetrieve` value. In this case, the cost of completing the search diminishes the performance benefits of retrieving fewer entries.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListEntryWithFields, ARGetOneEntryWithFields, ARCreateEntry, ARDeleteEntry, ARGetEntry, ARGetEntryStatistics, ARLoadARQualifierStruct, ARMergeEntry, ARSetEntry. See FreeAR for: `FreeAREntryListFieldList`, `FreeAREntryListList`, `FreeARQualifierStruct`, `FreeARSortList`, `FreeARStatusList`.

6.2.70 ARGetListEntryBlocks

Description

Retrieves a list of blocks of entries from the specified server. Data is returned as a data structure, `AREntryListBlock`. Entries are encapsulated in the `AREntryListBlock` data structure and divided into blocks of entries. Your program calls `ARGetEntryBlock` with a block number to return a list of entries for that block.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to entries the user can access (users must have permission for the **entryId** field to access and retrieve entries).

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListEntryBlock(
   ARControlStruct *control,
   ARNameType schema,
   ARQualifierStruct *qualifier,
   AREntryListFieldList *getListFields,
   ARSortList *sortList,
   unsigned int numRowsPerBlock,
   unsigned int firstRetrieve,
   unsigned int maxRetrieve,
   ARBoolean useLocale,
   AREntryBlockList *entryBlockList,
   unsigned int *numReturnedRows,
   unsigned int *numMatches,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form to retrieve entries for.

qualifier

A query that determines the set of entries to retrieve. The qualification can include one or more fields and any combination of conditional, relational, and arithmetic (numeric data types only) operations. The system generates an error if the user does not have read permission for a field or a field does not exist. Specify `NULL` or assign an operation value of `0` (`AR_COND_OP_NONE`) to match all form entries.

getListFields

A list of zero or more fields to be retrieved with each entry. The list can include any data fields except diary fields and long character fields. The system checks the permissions for each specified field and returns only those fields for which you have read access.

sortList

A list of zero or more fields that identifies the entry sort order. The system generates an error if you do not have read access on all specified fields. Specify `NULL` for this parameter (or zero fields) to use the default sort order for the form. The system sorts the entries in ascending order by **entryId** if the form has no default sort order.

numRowsPerBlock

The number of rows per block of data retrieved.

firstRetrieve

The first entry to retrieve. A value of `0` (`AR_START_WITH_FIRST_ENTRY`) represents the first entry. A value of `1` will skip the first entry.

maxRetrieve

The maximum number of entries to retrieve. Use this parameter to limit the amount of data returned if the qualification does not sufficiently narrow the list. Specify `0` (`AR_NO_MAX_LIST_RETRIEVE`) to assign no maximum.

useLocale

A flag that indicates whether to search for entries based on the locale. If you specify `1` (`TRUE`) and the Localize Server option is selected, entries are searched using the locale specified in `AR_RESERV_LOCALE_LOCALIZED_SCHEMA`. If no matches are found for the specified locale, the search becomes less restrictive until a match is found. If you specify `0` (`FALSE`) or the Localize Server option is cleared, all entries are searched. For more information, see Setting the Localize Server option.

Return values

entryBlockList

A list of zero or more (accessible) entries that match the criteria defined by the `qualifier` parameter. The system returns a list with zero items if no entries match the specified criteria.

numReturnedRows

The total number of returned rows in the block.

numMatches

The total number of (accessible) entries that match the qualification criteria. This value does not represent the number of entries returned unless the number of matching entries is less than or equal to the `maxRetrieve` value. Specify `NULL` for this parameter if you do not want to retrieve this value.

Note

Performing this count requires additional search time if the number of matching entries is more than the `maxRetrieve` value. In this case, the cost of completing the search diminishes the performance benefits of retrieving fewer entries.

status

A list of zero or more notes, warnings, or errors generated from a call to this function.

See also

ARGetEntryBlock.

6.2.71 ARGetListEntryWithFields

Description

Retrieves a list of form entries from the specified server. Data from each entry is returned as field/value pairs for all fields. You can limit the list to entries that match particular conditions by specifying the `qualifier` parameter. `ARGetListEntry` also returns a qualified list of entries, but as an unformatted string with a maximum length of 128 bytes for each entry containing the concatenated values of selected fields.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to entries the user can access (users must have permission for the **entryId** field to access and retrieve entries).

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListEntryWithFields(
   ARControlStruct *control,
   ARNameType schema,
   ARQualifierStruct *qualifier,
   AREntryListFieldList *getListFields,
   ARSortList *sortList,
   unsigned int firstRetrieve,
   unsigned int maxRetrieve,
   ARBoolean useLocale,
   AREntryListFieldValueList *entryList,
   unsigned int *numMatches,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form to retrieve entries for.

qualifier

A query that determines the set of entries to retrieve. The qualification can include one or more fields and any combination of conditional, relational, and arithmetic (numeric data types only) operations. The system generates an error if the user does not have read permission for a field or a field does not exist. Specify `NULL` or assign an operation value of `0` (`AR_COND_OP_NONE`) to match all form entries.

getListFields

A list of zero or more fields to be retrieved with each entry. The system checks the permissions for each specified field and returns only those fields for which you have read access.

sortList

A list of zero or more fields that identifies the entry sort order. The system generates an error if you do not have read access on all specified fields. Specify `NULL` for this parameter (or zero fields) to use the default sort order for the form (see ARCreateSchema). The system sorts the entries in ascending order by `entryId` if the form has no default sort order.

firstRetrieve

The first entry to retrieve. A value of `0` (`AR_START_WITH_FIRST_ENTRY`) represents the first entry. A value of `1` will skip the first entry.

useLocale

A flag that indicates whether to search for entries based on the locale. If you specify `1` (`TRUE`) and the Localize Server option is selected, entries are searched using the locale specified in `AR_RESERV_LOCALE_LOCALIZED_SCHEMA`. If no matches are found for the specified locale, the search becomes less restrictive until a match is found. If you specify `0` (`FALSE`) or the Localize Server option is cleared, all entries are searched. For more information, see Setting the Localize Server option.

maxRetrieve

The maximum number of entries to retrieve. Use this parameter to limit the amount of data returned if the qualification does not sufficiently narrow the list. Specify `0`(`AR_NO_MAX_LIST_RETRIEVE`) to assign no maximum.

Return values

entryList

A list of zero or more (accessible) entries that match the criteria defined by the `qualifier` parameter. The system returns a list with zero items if no entries match the specified criteria.

numMatches

The total number of (accessible) entries that match the qualification criteria. This value does not represent the number of entries returned unless the number of matching entries is less than or equal to the `maxRetrieve` value. Specify `NULL` for this parameter if you do not want to retrieve this value.

⚠

Note

Performing this count requires additional search time if the number of matching entries is more than the `maxRetrieve` value. In this case, the cost of completing the search diminishes the performance benefits of retrieving fewer entries.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListEntry, ARCreateEntry, ARDeleteEntry, ARGetEntry, ARGetEntryStatistics, ARLoadARQualifierStruct, ARMergeEntry, ARSetEntry. See FreeAR for: `FreeAREntryListFieldList`, `FreeAREntryListList`, `FreeARQualifierStruct`, `FreeARSortList`, `FreeARStatusList`.

6.2.72 ARGetListEntryWithMultiSchemaFields

Description

Performs dynamic joins by querying across multiple forms (including view and vendor forms) at run time. Depending on the structure of your data, this function might also perform the following suboperations to support dynamic joins:

- *Recursive queries* — If a form contains hierarchical (parent-child) data, such as manager and employee relationships, this function can use one query to search the form recursively and retrieve the complete hierarchy and related attributes. See Recursive queries.
- *Value set queries* — This function can perform `IN` and `NOT IN` operations on a fixed value set and on a value set returned by a subquery. See Value set queries.
- *Aggregate functions, group by clause, and having clause* — To refine the selection of data retrieved from the database and reduce the need for in-memory processing of large numbers of records, you can now apply aggregate functions along with group by and having clauses. (AR System releases prior to 7.6.02 do not support these options.)
- *Vendor form joins* — Joining vendor forms with other types of forms enables a single query to access data in both BMC Remedy AR System and external sources. See Using aggregate functions, groupBy clauses, and having clauses.

Data in each record is returned as multischema field/value pairs. You can limit the list to entries that match particular conditions by specifying the `qualifier` parameter.

For detailed structural information, see these documents:

- (C API) Structures for ARGetListEntryWithMultiSchemaFields
- (Java API) Javadoc comments

For examples of how to implement this function, see Retrieving entries from multiple forms.

Note

BMC Remedy Mid Tier does not support this function. You can create API programs, however, that use it. The programs must be able to parse the user-specified query and provide appropriate parameters to the function.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to entries that the user can access (users must have permission for the **entryId** field to access and retrieve entries).

Note

If both underlying forms of an BMC Remedy AR System join form contain the **Assignee Group** field (field ID 112), the join form inherits row-level access permissions from only the primary form. In queries across multiple forms, however, row-level access permissions for each form referenced in the query are evaluated separately; one form does not control access to the other.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListEntryWithMultiSchemaFields(
   ARControlStruct *control,
   ARMultiSchemaFuncQueryFromList *queryFromList,
   ARMultiSchemaFieldFuncList *getListFields,
   ARMultiSchemaQualifierStruct *qualifier,
   ARMultiSchemaSortList *sortList,
   unsigned int firstRetrieve,
   unsigned int maxRetrieve,
   ARBoolean useLocale,
   ARMultiSchemaFieldIdList *groupBy,
   ARMultiSchemaFuncQualifierStruct *having,
   ARMultiSchemaFieldFuncValueListList *entryList,
   unsigned int *numMatches,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where to perform the operation, and which session is used to perform it. The **user** and **server** fields are required.

queryFromList

A list of items to query and the join conditions between each item. An item can be a form (schema) or a recursive query. This parameter is used to generate the query's `FROM` clause.

All types of forms except display-only can be joined.

Only one recursive query can be included in the list, and it must be the first item in the list. The recursive query can operate on only one form in the list. (See ARMultiSchemaRecursiveFuncQueryStruct.)

Nested queries are not supported.

You can use inner, left outer, and right outer joins. If join information between objects in the list is not specified, the result set is a Cartesian product.

For more information, see ARMultiSchemaFuncQueryFromList.

getListFields

A list of zero or more fields to retrieve for each entry from the forms listed in the `queryFromList` parameter, along with the type of aggregate, if any. This parameter is used to generate the query's `SELECT` statement.

If no fields are specified, the fields in the Results List Fields property of each form in the `queryFromList` parameter are returned. If that property is empty, the form's Short Description field is returned.

The system checks the permissions for each field and returns only those fields for which the user has read access.

Because `ARGetListEntryWithMultiSchemaFields` can operate on multiple forms, each field in this list must be qualified by a form name or alias. Additionally, each field in this list must specify the aggregate function being applied to the field.

`getListFields` supports the following aggregate functions:

- `NONE`
- `SUM`
- `AVG`
- `COUNT`
- `MIN`
- `MAX`

Note

As of release 7.6.02, `getListFields` is an `ARMultiSchemaFieldFuncList` structure. In earlier releases, it was an `ARMultiSchemaFieldIDList` structure. `ARMultiSchemaFieldIDList` does not support aggregate functions. If you send an `ARGetListEntryWithMultiSchemaFields` call that specifies aggregate functions to a pre-7.6.02 BMC Remedy AR System server, the new options are stripped from the query.

See ARMultiSchemaFieldFuncStruct, ARMultiSchemaFieldFuncList, and ARMultiSchemaFuncQualifierStruct.

qualifier

Zero or more conditions that limit the set of entries that this function retrieves. This parameter is used to generate the query's `WHERE` clause. (For some databases, the join qualifiers are also used to generate the `WHERE` clause.)

A qualification can include one or more fields and any combination of conditional, relational, arithmetic (numeric data types only), and `IN` or `NOT IN` operations. `IN` and `NOT IN` operations can be performed on a fixed value set or on a value set returned by a subquery.

The system generates an error if the user does not have read permission for a field or if a field does not exist. To retrieve all entries, set this parameter to `NULL` or set its `operation` element to `0` (`AR_COND_OP_NONE`).

See ARMultiSchemaQualifierStruct.

sortList

A list of zero or more fields that specifies the entry sort order. This parameter is used to generate the primary query's `ORDER BY` clause. This parameter cannot be used in subqueries.

If no field is specified, the system sorts the entries according to the first form's default sort order. The system generates an error if you do not have read access to all specified fields.

See ARMultiSchemaSortList.

Note

For recursive queries, set the sortList parameter to `NULL`. See Recursive queries.

firstRetrieve

The first entry to retrieve. A value of `0` (`AR_START_WITH_FIRST_ENTRY`) represents the first entry. A value of `1` will skip the first entry.

maxRetrieve

The maximum number of entries to retrieve. Use this parameter to limit the amount of data returned if the qualification does not sufficiently narrow the list. To assign no maximum, specify `0` (`AR_NO_MAX_LIST_RETRIEVE`).

useLocale

A flag that indicates whether to search for entries based on the locale. If you specify `1` (`TRUE`) and the Localize Server option is selected, entries are searched using the locale specified in `AR_RESERV_LOCALE_LOCALIZED_SCHEMA`. If no matches are found for the specified locale, the search becomes less restrictive until a match is found. If you specify `0` (`FALSE`) or the Localize Server option is cleared, all entries are searched. For more information, see Setting the Localize Server option.

groupBy

A list of zero or more fields used to partition the result set. This parameter is used to generate the primary query's `GROUP BY` clause. This field can be null. If no fields are specified, no `GROUP BY` clause is generated.

A common use for `GROUP BY` is to create partitions in the result set over which the database server computes an aggregate function. For example, consider a table of ticket IDs, ticket statuses, and agent names. A SQL query to discover for each agent how many tickets are open would look like this:

```
SELECT COUNT(ticketId),agent FROM tickets WHERE status='Open' GROUP BY agent
```

Note

If you send an `ARGetListEntryWithMultiSchemaFields` call with a non-empty `groupBy` list to a pre-7.6.0.2 AR System server, the `GROUP BY` clause is stripped from the query.

The following rules apply to the `groupBy` parameter:

- If `groupBy` is specified, all fields in the select list must be aggregates or included in the `groupBy` list.
- Aggregate fields and sub-queries (selects) might not appear in the `groupBy` list.
- Fields might appear in the `groupBy` list that are not in the select list.
- Fields used in the `groupBy` list do not have to be in the `having` list and fields used in the `having` list do not need to be in the `groupBy` list.

having

Used to generate the primary query's `HAVING` clause. This field can be null. If null or an empty qualifier is specified, no `HAVING` clause is generated.

The `HAVING` clause filters the result set. Unlike the qualifier, it can specify aggregate functions. For example, to modify the example query used to illustrate the `groupBy` clause to further limit the result list to agents with more than 10 tickets open, the query would look like this:

```
SELECT COUNT(ticketId),agent FROM tickets WHERE status='Open' GROUP BY agent HAVING
COUNT(ticketId) > 10
```

Note

If you send an `ARGetListEntryWithMultiSchemaFields` call with a non-empty `having` qualifier to a pre-7.6.0.2 AR System server, the HAVING clause is stripped from the query.

The following rules apply to the `having` parameter:

- Numeric or enumerated fields must be used for any function other than `COUNT`.
- Aggregate and non-aggregate fields can be in the `having` qualifier.
- Fields might appear in the `having` qualifier that are not in the select list.
- Fields used in the `groupBy` list do not have to be in the `having` clause and fields used in the `having` clause do not need to be in the `groupBy` list.

Return values

entryList

A list of zero or more (accessible) entries that match the criteria specified in the `qualifier` parameter. The system returns an `ARMultiSchemaFieldFuncValueListList`, in which each field contains the `queryAlias`, `fieldId`, and `funcID`. The list contains zero items if no entries match the specified criteria.

numMatches

The total number of (accessible) entries that match the qualification criteria. This value does not represent the number of entries returned unless the number of matching entries is less than or equal to the `maxRetrieve` value. If you do not want to retrieve this value, set this parameter to `NULL`.

Note

Performing this count requires additional search time if the number of matching entries is more than the `maxRetrieve` value. In this case, the cost of completing the search diminishes the performance benefits of retrieving fewer entries.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.73 ARGetListEscalation

Description

Retrieves a list of escalation names from the specified server. You can retrieve all escalations or limit the list to escalations associated with a particular form. The function returns all escalations modified on or after the timestamp.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListEscalation(
   ARControlStruct *control,
   ARNameType schema,
   ARTimestamp changedSince,
   ARPropList *objPropList,
   ARNameList *nameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

schema

The name of the form to retrieve escalations for. Specify `NULL` for this parameter to retrieve escalations for all forms.

changedSince

A time stamp that limits the escalation retrieved to those modified after the specified time. Specify **0** for this parameter to retrieve escalations with any modification time stamp.

objPropList

List of server-managed object properties to search for. Returns all escalations that match the object properties, for example, the application owner. See Server-managed object property tags.

Return values

nameList

A list of zero or more escalations that match the criteria defined in the `changedSince` and `objPropList` arguments. The system returns a list with zero items if no forms match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEscalation, ARDeleteEscalation, ARDeleteSchema, ARGetEscalation, ARSetEscalation. See FreeAR for: `FreeARNameList`, `FreeARStatusList`.

6.2.74 ARGetListExtSchemaCandidates

Description

Retrieves a list of all available external data source tables (schema candidates). Users select fields from these candidates to populate the vendor form.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to forms the user can access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListExtSchemaCandidates(
   ARControlStruct *control,
   unsigned int schemaType,
   ARCompoundSchemaList *schemaList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schemaType

A value that indicates the form types to retrieve.

Bit 7:	Retrieve all view schemas (`AR_SCHEMA_VIEW`).
Bit 8:	Retrieve all vendor schemas (`AR_SCHEMA_VENDOR`).

To retrieve both visible and hidden forms, add `1024` (`AR_HIDDEN_INCREMENT`) to the form type. For example, specify `AR_SCHEMA_VENDOR | AR_HIDDEN_INCREMENT` for this parameter.

Return values

schemaList

A list of zero or more (accessible) forms that match the criteria in the `schemaType` parameter. Specify `NULL` for this parameter if you do not want to retrieve the list.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetMultipleExtFieldCandidates. See FreeAR for: `FreeARCompuondSchemaList`, `AFreeARStatusList`.

6.2.75 ARGetListField

Description

Retrieves a list of field IDs for a particular form from the specified server. You can retrieve all fields or limit the list to fields of a particular type or fields modified after a specified time.

Privileges

This operation can be performed by users with access permission for the specified form.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListField(
   ARControlStruct *control,
   ARNameType schema,
   unsigned long fieldType,
   ARTimestamp changedSince,
   ARPropList objPropList,
   ARInternalIdList *idList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the AR_SESS_CONTROL_PROP_API_OVERLAYGROUP variable of the ARSetSessionConfiguration function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the AR_SESS_CONTROL_PROP_API_OVERLAYGROUP variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

schema

The name of the form to retrieve fields for.

fieldType

A bitmask that indicates the field types to retrieve.

Bit 0:	Retrieve data fields (AR_FIELD_TYPE_DATA).
Bit 1:	Retrieve trim fields (AR_FIELD_TYPE_TRIM).
Bit 2:	Retrieve control fields (AR_FIELD_TYPE_CONTROL).
Bit 3:	Retrieve panel fields (AR_FIELD_TYPE_PAGE).
Bit 4:	Retrieve panel holder fields (AR_FIELD_TYPE_PAGE_HOLDER).
Bit 5:	Retrieve table fields (AR_FIELD_TYPE_TABLE).
Bit 6:	

	Retrieve column fields (`AR_FIELD_TYPE_COLUMN`).
Bit 8:	Retrieve vendor type fields (`AR_FIELD_TYPE_VENDOR`).
Bit 128:	Retrieve attachment type fields (`AR_FIELD_TYPE_ATTACH`).
Bit 256:	Retrieve attachment pool type fields (`AR_FIELD_TYPE_ATTACH_POOL`).

changedSince

A time stamp that limits the fields retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve fields with any modification time stamp.

objPropList

List of server-managed object properties to search for. Returns all images that match the object properties.

Return values

idList

A list of zero or more fields that match the criteria defined by the `fieldType` and `changedSince` parameters. The system returns a list with zero items if no fields match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateField, ARDeleteField, ARGetField, ARGetMultipleFields, ARSetField. See FreeAR for: `FreeARInternalIdList`, `FreeARStatusList`.

6.2.76 ARGetListFilter

Description

Retrieves a list of filter names from the specified server. You can retrieve all filters or limit the list to filters associated with a particular form or modified after a specified time.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListFilter(
   ARControlStruct *control,
   ARNameType schema,
   ARTimestamp changedSince,
   ARPropList *objPropList,
   ARNameList *nameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

schema

The name of the form to retrieve filters for. Specify `NULL` for this parameter to retrieve filters for all forms.

changedSince

A time stamp that limits the filters retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve filters with any modification time stamp.

objPropList

List of server-managed object properties to search for. Returns all filters that match the object properties, for example, the application owner. See Server-managed object property tags.

Return values

nameList

A list of zero or more (accessible) filters that match the criteria defined by the `changedSince` and `objPropList` parameters. The system returns a list with zero items if no filters match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateFilter, ARDeleteFilter, ARDeleteSchema, ARGetFilter, ARSetFilter. See FreeAR for: `FreeARNameList`, `FreeARStatusList`.

6.2.77 ARGetListGroup

Description

Retrieves a list of access control groups from the specified server. You can retrieve all groups or limit the list to groups associated with a particular user.

Privileges

Group information for the current user can be retrieved by all users. Access to group information for other users is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListGroup(
   ARControlStruct *control,
   ARAccessNameType userName,
   ARAccessNameType password,
   ARGroupInfoList *groupList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

userName

The name of the user for which the system retrieves group information. Specify `NULL` for this parameter to retrieve all groups on the server.

password

The password of the user for whom the system retrieves the list. If this parameter is `NULL`, the function returns the group list for the first user whose name matches the name passed in, not necessarily the user who requested the list. If this parameter specifies a password, the server returns the group list for the user whose name and password match those in the parameter list.

Return values

groupList

A list of zero or more groups that match the criteria defined by the `userName` parameter. Each item in the list contains a group ID, the name associated with that ID, group category, and the computed group qualification string. The system returns a list with zero items if no groups match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListUser. See FreeAR for: `FreeARGroupInfoList`, `FreeARStatusList`.

6.2.78 ARGetListImage

Description

Retrieves a list of image names from the specified server. You can retrieve all images or limit the list to those images associated with particular schemas, those modified after a specified time, and those of a specific type.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListImage(
```

```
ARControlStruct *control,
   ARNameList *schemaList,
   ARTimestamp changedSince,
   char *imageType,
   ARPropList objPropList,
   ARNameList *imageList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

schemaList

A list of names of forms. Only images associated with the named forms are retrieved. Specify `NULL` for this parameter to retrieve all images.

changedSince

A time stamp that limits the images retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve images with any modification time stamp.

imageType

Image type. Only images of the specified type are retrieved.

objPropList

List of server-managed object properties to search for. Returns all images that match the object properties.

Return values

imageList

A list of zero or more name of images that match the criteria defined by the `schemaList`, `changedSince`, and `imageType` parameters. The list does not contain duplicate names or the name of images that have been deleted. The list contains zero items if no images match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateImage, ARDeleteImage, ARGetImage, ARGetMultipleImages, ARSetImage. See FreeAR for: `FreeARNameList`, `FreeARStatusList`.

6.2.79 ARGetListLicense

Description

Retrieves a list of entries from the BMC Remedy AR System License form for the specified server. The list can contain information for all types of licenses, including DSO, flashboards, SMU applications, and servers.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListLicense(
   ARControlStruct *control,
   ARLicenseNameType licenseType,
   ARLicenseInfoList *licenseInfoList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

licenseType

The type of license to retrieve. If the value of this argument is `NULL`, the function retrieves all the license types that are available.

Return values

licenseInfoList

A list of licenses that fit the specified criteria. For each license, information such as license key, license type, expiration date, number of licenses, and qualifier is included. If the Add or Remove Licenses form contains information not used by BMC Remedy AR System, blank fields are returned in place of the unused information.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateLicense, ARDeleteLicense, ARValidateLicense, ARValidateMultipleLicenses.

6.2.80 ARGetListRole

Description

Retrieves a list of roles for a deployable application or returns a list of roles for a user for a deployable application.

Privileges

Role information for the current user can be retrieved by all users. Access to role information for other users is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListRole(
ARControlStruct *control,
   ARNameType applicationName,
   ARAccessNameType userName,
   ARAccessNameType password,
   ARRoleInfoList *roleList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

applicationName

The name of the specific application to get roles from.

userName

The name of the specific user for which the system retrieves roles information. Specify `NULL` to retrieve all roles for the application.

password

The password of the specific user for whom the system retrieves the list. If this parameter is `NULL`, the function returns the role list for the first user whose name matches the name passed in, not necessarily the user who requested the list. If this parameter specifies a password, the server returns the role list for the user whose name and password match those in the parameter list.

Return values

roleList

A list of zero or more roles that match the criteria defined by the `userName` parameter. Each item in the list contains a role ID, the name associated with that ID, and the group mapping. The system returns a list with zero items if no roles match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListGroup.

6.2.81 ARGetListSchema

Description

Retrieves a list of forms from the specified server. You can retrieve all (accessible) forms or limit the list to forms of a particular type or forms modified after a specified time.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to forms the user can access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListSchema(
   ARControlStruct *control,
   ARTimestamp changedSince,
   unsigned int schemaType,
   ARNameType name,
   ARInternalIdList *fieldIdList,
   ARPropList *objPropList,
   ARNameList *nameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the forms retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve forms with any modification time stamp.

schemaType

A value that indicates the form types to retrieve (some types are mutually exclusive).

`0:`	Retrieve all form types (AR_LIST_SCHEMA_ALL).
`1:`	Retrieve all base forms (AR_LIST_SCHEMA_REGULAR).
`2:`	Retrieve all join forms (AR_LIST_SCHEMA_JOIN).
`3:`	Retrieve all view forms (AR_LIST_SCHEMA_VIEW).

`4:`	Retrieve all join forms that depend on the form specified by the name parameter (AR_LIST_SCHEMA_UPLINK).
`5:`	Retrieve all base or join forms that the form specified by the name parameter depends on (AR_LIST_SCHEMA_DOWNLINK).
`6:`	Retrieve all display-only forms (AR_LIST_SCHEMA_DIALOG).
`7:`	Retrieve all forms with data fields (AR_LIST_SCHEMA_ALL_WITH_DATA).
`8:`	Retrieve all vendor schemas (AR_LIST_SCHEMA_VENDOR).

To retrieve both visible and hidden forms, add `1024` (`AR_HIDDEN_INCREMENT`) to the form type. For example, specify `AR_LIST_SCHEMA_ALL | AR_HIDDEN_INCREMENT` for this parameter.

name

If the `schemaType` is `AR_LIST_SCHEMA_UPLINK`, this parameter specifies the form on which the retrieved forms depend. If the `schemaType` is `AR_LIST_SCHEMA_DOWNLINK`, this parameter specifies the form that depends on the retrieved forms. This parameter is ignored if you do not specify `AR_LIST_SCHEMA_UPLINK` or `AR_LIST_SCHEMA_DOWNLINK` as the `schemaType`.

fieldIdList

List of form fields. The system returns only the forms that contain all the fields in this list. Specify `NULL` for this parameter (or zero fields) if you do not want to qualify the forms returned by the fields they contain. For example, specify the four reserved server event fields (`800`, `801`, `802`, `803`) in this list to retrieve the server event schema.

Note

Archive forms are not returned if you specify a list of form fields. You must specify `NULL` to return archive forms.

objPropList

List of server-managed object properties to search for. Returns all schemas that match the object properties, for example, the application owner. See Server-managed object property tags.

Return values

nameList

A list of zero or more (accessible) forms that match the criteria in the `schemaType`, `changedSince`, `fieldIdList`, and `objPropList` parameters. The system returns a list with zero items if no forms match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateSchema, ARDeleteSchema, ARGetListSchemaWithAlias, ARGetSchema, ARSetSchema. See FreeAR for: `FreeARNameList`, `FreeARStatusList`, `FreeARInternalIDList`.

6.2.82 ARGetListSchemaWithAlias

Description

Retrieves a list of form definitions and their corresponding aliases from the specified server. You can retrieve all (accessible) forms or limit the list to forms of a particular type or forms modified after a specified time.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to forms the user can access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListSchemaWithAlias(
   ARControlStruct *control,
   ARTimestamp changedSince,
   unsigned int schemaType,
   ARNameType name,
   ARInternalIdList *fieldIdList,
   ARNameType vuiLabel,
   ARPropList *objPropList,
   ARNameList *nameList,
   ARNameList *aliasList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the AR_SESS_CONTROL_PROP_API_OVERLAYGROUP variable of the ARSetSessionConfiguration function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the AR_SESS_CONTROL_PROP_API_OVERLAYGROUP variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the forms retrieved to those modified after the specified time. Specify 0 for this parameter to retrieve schemas with any modification time stamp.

schemaType

A value that indicates the form types to retrieve (some types are mutually exclusive).

0:	Retrieve all form types (AR_LIST_SCHEMA_ALL).
1:	Retrieve all base forms (AR_LIST_SCHEMA_REGULAR).
2:	Retrieve all join forms (AR_LIST_SCHEMA_JOIN).
3:	Retrieve all view forms (AR_LIST_SCHEMA_VIEW).
4:	Retrieve all join forms that depend on the form specified by the name parameter (AR_LIST_SCHEMA_UPLINK).
5:	Retrieve all base or join forms that the form specified by the name parameter depends on (AR_LIST_SCHEMA_DOWNLINK).
6:	Retrieve all display-only forms (AR_LIST_SCHEMA_DIALOG).
7:	Retrieve all forms with data fields (AR_LIST_SCHEMA_ALL_WITH_DATA).
8:	Retrieve all vendor schemas (AR_SCHEMA_VENDOR).

To retrieve both visible and hidden forms, add 1024 (AR_HIDDEN_INCREMENT) to the form type. For example, specify AR_LIST_SCHEMA_ALL | AR_HIDDEN_INCREMENT for this parameter.

name

If the schemaType is AR_LIST_SCHEMA_UPLINK, this parameter specifies the form on which the retrieved forms depend. If the schemaType is AR_LIST_SCHEMA_DOWNLINK, this parameter specifies the form that depends on the retrieved forms. This parameter is ignored if you do not specify AR_LIST_SCHEMA_UPLINK or AR_LIST_SCHEMA_DOWNLINK as the schemaType.

fieldIdList

A list of zero or more internal IDs that specify the fields to retrieve. Specify `NULL` for this parameter (or zero fields) if you do not want to qualify the forms returned by the fields they contain. For example, specify the four reserved server event fields (`800`, `801`, `802`, `803`) in this list to retrieve the server event schema.

vuiLabel

Label for the specific VUI from which to retrieve aliases. The system returns aliases from the default VUI if you specify `NULL` or if the specified VUI does not exist.

objPropList

List of server-managed object properties to search for. Returns all schemas that match the object properties, for example, the application owner. See Server-managed object property tags.

Return values

nameList

A list of zero or more (accessible) schemas that match the criteria defined in the `schemaType`, `changedSince`, `fieldIDList`, and `objPropList` parameters. The system returns a list with zero items if no forms match the specified criteria.

aliasList

A list of zero or more VUI aliases that:

- Match the schemas in the `nameList` parameter.
- Are aliases from the form that the `name` parameter specifies.
- Conform to the locale that the `control` parameter specifies.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateSchema, ARDeleteSchema, ARGetListSchema, ARGetSchema, ARSetSchema. See FreeAR for: `FreeARInternalIdList`, `FreeARNameList`, `FreeARStatusList`.

6.2.83 ARGetListServer

Description

Retrieves the list of available BMC Remedy AR System servers defined in the **ar** directory file (UNIX only). BMC Remedy Developer Studio and BMC Remedy Data Import connect to these servers automatically if no servers are specified at startup. If the **ar** file is under NIS control, the system uses the file specified by the NIS map instead of the local **ar** file. For information about the **ar** file, see ar.cfg or ar.conf.

Note

In the Windows API, server information is retrieved from the registry instead of the **ar** file. API programs that run on the server (for example, through a filter or escalation) can use this function to retrieve only the name of that local server. Programs that run on a Windows client, however, cannot. In this case, the function always returns a list of zero servers.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARGetListServer(
   ARControlStruct *control,
   ARServerNameList *serverList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

serverList

A list of zero or more registered BMC Remedy AR System servers. The system returns a list with zero items if no BMC Remedy AR System servers are registered.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ar directory file. See FreeAR for: `FreeARServerNameList`, `FreeARStatusList`.

6.2.84 ARGetListSQL

Description

Retrieves a list of rows from the underlying SQL database on the specified server. The server executes the SQL command that you specify and returns the matching rows. A list with zero items and a warning message are returned if no SQL database resides on the server. The system returns information based on the access privileges of the user who launched the BMC Remedy AR System server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListSQL(
   ARControlStruct *control,
   char *sqlCommand,
   unsigned int maxRetrieve,
   ARValueListList *valueListList,
   unsigned int *numMatches,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

sqlCommand

The SQL command to execute (following the syntax rules for the underlying database). The owner of the BMC Remedy AR System server process must have permission to perform the specified SQL operation.

maxRetrieve

The maximum number of rows to retrieve. Use this parameter to limit the amount of data returned if the SQL query does not sufficiently narrow the list. Specify `0` (`AR_NO_MAX_LIST_RETRIEVE`) to assign no maximum.

Return values

valueListList

A list of zero or more (accessible) rows that match the criteria defined by the `sqlCommand` parameter. Each item in the list represents one matching row, each of which contains a list of the selected column values. The system returns a list with zero items if no rows match the specified criteria.

numMatches

The total number of (accessible) rows that match the SQL selection criteria. This value does not represent the number of rows returned unless the number of matching rows is less than or equal to the `maxRetrieve` value. Specify `NULL` for this parameter if you do not want to retrieve this value.

Note

Performing this count requires additional search time if the number of matching rows is more than the `maxRetrieve` value. In this case, the cost of completing the search diminishes the performance benefits of retrieving fewer rows.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARExecuteProcess, ARGetListEntry. See FreeAR for: `FreeARStatusList`, `FreeARValueListList`.

6.2.85 ARGetListSupportFile

Description

Retrieves a list of support file IDs for a specified type of object.

Privileges

Any user who has access to the specified object.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListSupportFile(
   ARControlStruct *control,
   unsigned int fileType,
   ARNameType name,
   ARInternalId id2,
   ARTimestamp changedSince,
   ARInternalIdList *fileIdList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

fileType

The numerical value for the type of file, and the type of object the file is related to. Specify `1` (`AR_SUPPORT_FILE_EXTERNAL_REPORT`) for an external report file associated with an active link.

name

The name of the object the file is associated with, usually a form.

id2

The ID of the field or VUI, if the object is a form. If the object is not a form, set this parameter to `0`.

changedSince

A time stamp that limits the IDs retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve all IDs.

Return values

fileIdList

A list of support file IDs linked to an object.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent, ARCreateSupportFile, ARDeleteActiveLink, ARDeleteSupportFile, ARGetActiveLink, ARGetSupportFile, ARSetActiveLink, ARSetSupportFile.

6.2.86 ARGetListUser

Description

Retrieves a list of users from the specified BMC Remedy AR System server. You can retrieve information about the current user, all registered users, or all users currently accessing the server.

Privileges

Information about the current user can be retrieved by all users. Access to information about other users is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListUser(
   ARControlStruct *control,
   unsigned int userListType,
   ARTimestamp changedSince,
   ARUserInfoList *userList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

userListType

A value that indicates the user type to retrieve.

0:	Retrieve the current user (`AR_USER_LIST_MYSELF`).
1:	Retrieve all registered users (`AR_USER_LIST_REGISTERED`).
2:	Retrieve all users currently accessing the server (`AR_USER_LIST_CURRENT`).
3:	Retrieve all users with accounts marked invalid (`AR_USER_LIST_INVALID`) .
4:	Retrieve all users with application licenses (`AR_USER_LIST_APPLICATION`).

changedSince

A time stamp that limits the list entries retrieved to those modified after the specified time. Specify 0 for this parameter to retrieve entries with any modification time stamp. The `changedSince` parameter applies only if the `userListType` parameter is set to: `AR_USER_LIST_REGISTERED`.

Return values

userList

List of zero or more users that match the criteria defined by the `userListType` parameter. The list contains the user name and license type. If you retrieve the current user, the list also contains a time stamp that specifies the last server access. The system returns a list with zero items if no users match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListGroup. See FreeAR for: `FreeARUserInfoList, FreeARStatusList`.

6.2.87 ARGetListVUI

Description

Retrieves a list of form views (VUI) for a particular form on the specified server. You can retrieve all views or limit the list to views modified after a specified time.

Privileges

This operation can be performed by users with access permission for the specified form.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetListVUI(
   ARControlStruct *control,
   ARNameType schema,
   ARTimestamp changedSince,
   ARPropList objPropList,
   ARInternalIdList *idList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

schema

The name of the form to retrieve views for.

changedSince

A time stamp that limits the views retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve views with any modification time stamp.

objPropList

List of server-managed object properties to search for. Returns all images that match the object properties.

Return values

idList

A list of zero or more view IDs that match the criteria defined by the `changedSince` parameter. The system returns a list with zero items if no views match the specified criteria.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateVUI, ARDeleteVUI, ARGetVUI, ARSetVUI. See FreeAR for: `FreeARInternalIdList`, `FreeARStatusList`.

6.2.88 ARGetLocalizedValue

Description

Retrieves a localized text string from the BMC Remedy Message Catalog. The message that the server retrieves depends on the user locale.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetLocalizedValue(
   ARControlStruct *control,
   ARLocalizedRequestStruct *localizedRequest,
   ARValueStruct *localizedValue,
   ARTimestamp *timestamp,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

localizedRequest

Identification for the localized value to be retrieved from the BMC Remedy Message Catalog. It contains the type of value, value name, and, for some types of values, unique identification numbers.

Return values

localizedValue

The localized value to be retrieved. The value can be a character value or an attachment. Specify `NULL` for this argument if you do not want to retrieve this value.

timestamp

A time stamp that specifies the last change to the value. Specify `NULL` for this parameter if you do not want to retrieve this value.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetMultipleLocalizedValues. See FreeAR for: `FreeARValueStruct`, `FreeARStatusList`.

6.2.89 ARGetMultipleActiveLinks

Description

Retrieves multiple active link definitions. This function performs the same action as `ARGetActiveLink` but is easier to use and more efficient than retrieving information about multiple active links one by one.

Information is returned in lists for each item, with one item in the list for each active link returned. For example, if the second item in the list for `existList` is `TRUE`, the name of the second active link is returned in the second item in the list for `actLinkNameList`.

Privileges

All users who have permission for the active link. Only BMC Remedy AR System administrators have access to `groupList` information.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleActiveLinks(
   ARControlStruct *control,
   ARTimestamp changedSince,
   ARNameList *nameList,
   ARBooleanList *existList,
   ARNameList *actLinkNameList,
   ARUnsignedIntList *orderList,
   ARWorkflowConnectList *schemaList,
   ARInternalIdListList *assignedGroupListList,
   ARInternalIdListList *groupListList,
   ARUnsignedIntList *executeMaskList,
   ARInternalIdList *controlFieldList,
   ARInternalIdList *focusFieldList,
   ARUnsignedIntList *enableList,
   ARQualifierList *queryList,
   ARActiveLinkActionListList *actionListList,
   ARActiveLinkActionListList *elseListList,
   ARTextStringList *helpTextList,
   ARTimestampList *timestampList,
   ARAccessNameList *ownersList,
   ARAccessNameList *lastChangedList,
   ARTextStringList *changeDiaryList,
   ARPropListList *objPropListList,
   ARUnsignedIntList *errorActlinkOptionsList,
   ARNameList *errorActlinkNameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the active links retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve active links with any modification time stamp.

nameList

A list of names of the active links to retrieve that match the criteria in the `changedSince` argument. The system returns a list with zero items if no active links match the specified criteria.

Note

If you enter the same active link name two or more times in the nameList parameter, the server retrieves information for only the first occurrence, and ignores the information for the other repeat occurrences.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the active links exist and meet the qualifying criteria. Values: `TRUE` means the active link exists; `FALSE` means the active link does not exist.

actLinkNameList

A list of names of active links that this function returns. This argument returns names in the same order as the input list of active links to be retrieved. If namelist is not `NULL`, each return list maintains a corresponding position for each supplied name. As a consequence, you must see the existList returned to see if the active link in that position had any data returned for it.

orderList

A value between `0` and `1000` (inclusive) that determines the active link execution order. When multiple active links are associated with a form, the value associated with each active link determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

schemaList

The list of form names the active link is linked to. The active link must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

assignedGroupListList

A list of zero or more groups that are directly assigned permission to the active links.

groupListList

A list of zero or more groups who can access these active links, including the inheritance hierarchy in the case of hierarchical groups. Access to this information is limited to users with BMC Remedy AR System administrator privileges. Specify `NULL` for this parameter if you do not want to retrieve this value.

If the static inheritance property is not enabled for the active link, this list will be the same as the `assignedGroupListList`.

executeMaskList

A list of bitmasks that indicate the form operations that trigger the active links. See ARCreateActiveLink for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this value.

controlFieldList

A list of the IDs of the fields that represent the button, toolbar button, or menu items associated with executing the active links. The system returns zero if the `executeMask` does not include the `AR_EXECUTE_ON_BUTTON` condition. Specify `NULL` for this parameter if you do not want to retrieve this value.

focusFieldList

A list of the IDs of the fields associated with executing the active links by pressing Return or selecting a character menu item. The system returns zero if the `executeMask` does not include the `AR_EXECUTE_ON_RETURN or AR_EXECUTE_ON_MENU_CHOICE` conditions. Specify `NULL` for this parameter if you do not want to retrieve the focus fields.

enableList

A list of flags that specify whether the active link is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve these flags.

queryList

A list of qualifications that determines whether the active links are executed. The system returns zero (`AR_COND_OP_NONE`) if the active links have no qualifications. Specify `NULL` for this parameter if you do not want to retrieve the queries.

actionListList

A list of the sets of actions performed if the condition defined by the `query` parameter is satisfied. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve the action lists.

elseListList

A list of the sets of actions performed if the conditions defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve the else list.

helpTextList

A list of the help texts associated with the active links. Specify `NULL` for this parameter if you do not want to retrieve the help texts (which is useful if you are calling this function to verify whether instances of these objects exist).

timestampList

A list of time stamps that identify the last changes to the active links. Specify `NULL` for this parameter if you do not want to retrieve this value.

ownerList

A list of owners for the active links. Specify `NULL` for this parameter if you do not want to retrieve the owners.

lastChangedList

A list of users who made the last changes to the active links. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiaryList

A list of the change diaries associated with the active link. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to `NULL`, a properties list with zero properties is associated with the object, and when an `ARGetFilter` action is performed, zero properties are returned. See Server object properties.

errorActlinkOptionsList

Reserved for future use. Set to `NULL`.

errorActlinkNameList

Reserved for future use. Set to `NULL`.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateActiveLink, ARDeleteActiveLink, ARGetActiveLink, ARGetListActiveLink, ARGetMultipleEntries, ARGetMultipleExtFieldCandidates, ARGetMultipleFields, ARGetMultipleLocalizedValues, ARGetMultipleSchemas, ARSetActiveLink. See FreeAR for: `FreeARAccessNameList`, `FreeARActiveLinkActionListList`, `FreeARBooleanList`, `FreeARInternalIdList`, `FreeARInternalIdListList`, `FreeARNameList`, `FreeARPropListList`, `FreeARQualifierList`, `FreeARStatusList`, `FreeARTextStringList`, `FreeARTimestampList`, `FreeARUnsignedIntList`, `FreeARWorkflowConnectList`.

6.2.90 ARGetMultipleCharMenus

Description

Retrieves from the specified server information about the character menus whose names are specified by the `nameList` parameter. This function performs the same action as `ARGetCharMenu` but is easier to use and more efficient than retrieving information about multiple character menus one by one. Information is returned in lists for each item, with one item in the list for each menu returned. For example, if the second item in the list for `existList` is `TRUE`, the name of the second character menu is returned in the second item in the list for `charMenuNameList`.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleCharMenus(
   ARControlStruct *control,
   ARTimestamp changedSince,
   ARNameList *nameList,
   ARBooleanList *existList,
   ARNameList *charMenuNameList,
   ARUnsignedIntList *refreshCodeList,
   ARCharMenuStructList *menuDefnList,
   ARTextStringList *helpTextList,
   ARTimestampList *timestampList,
   ARAccessNameList *ownerList,
   ARAccessNameList *lastChangedList,
   ARTextStringList *changeDiaryList,
   ARPropListList *objPropListList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the character menus retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve character menus with any modification time stamp.

nameList

The names of the character menus to retrieve. The nameList can be passed as a `NULL` or as an empty list, as in:

```
ARNameList emptyNameList = {0, 0};
   ...
   ARGetMultipleCharMenus(control, changedSince, &emptyNameList, ...)
```

In this case information is returned for every character menu that passes the `changedSince` criterion.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the character menus exist. Values: `TRUE` means the character menu exists; `FALSE` means the character menu does not exist.

charMenuNameList

A list of character menu names retrieved.

refreshCodeList

A value that indicates when the list of character menus is refreshed. See ARCreateCharMenu for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this value.

menuDefnList

A list of definitions of character menus. Specify `NULL` for this parameter if you do not want to retrieve this value.

helpTextList

A list of help text items associated with the character menus. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of these objects exists).

timeStampList

A list of time stamps that specify the last change to the character menus. Specify `NULL` for this parameter if you do not want to retrieve this value.

ownerList

A list of owning users for the character menus. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChangedList

A list of users who made the last change to the character menus. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiaryList

A list of change diaries associated with the character menus. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary list. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropListList

A list of server object property lists that correspond to the menus in `nameList*`.* If this parameter is set to `NULL`, no properties are returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDeleteCharMenu, ARExpandCharMenu, ARGetCharMenu, ARGetListCharMenu, ARSetCharMenu. See FreeAR for: `FreeARCharMenuStruct`, `FreeARStatusList`, `FreeARPropList`.

6.2.91 ARGetMultipleContainers

Description

Retrieves multiple container objects.

Information is returned in lists for each item, with one item in the list for each container returned. For example, if the second item in the list for `existList` is `TRUE`, the name of the second container is returned in the second item in the list for `containerNameList`.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARGetMultipleContainers (
   ARControlStruct *control,
   ARTimestamp changedSince,
   ARNameList *nameList,
   ARContainerTypeList *containerTypes,
   unsigned int attributes,
   ARContainerOwnerObjList *ownerObjList,
   ARReferenceTypeList *refTypes,
   ARBooleanList *existList,
   ARNameList *containerNameList,
   ARPermissionListList *assignedGroupListList,
   ARPermissionListList *groupListList,
   ARInternalIdListList *admingrpListList,
   ARContainerOwnerObjListList *ownerObjListList,
   ARTextStringList *labelList,
   ARTextStringList *descriptionList,
   ARUnsignedIntList *typeList,
   ARReferenceListList *referenceList,
   ARTextStringList *helpTextList,
   ARAccessNameList *ownerList,
   ARTimestampList *timestampList,
   ARAccessNameList *lastChangedList,
   ARTextStringList *changeDiaryList,
   ARPropListList *objPropListList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the Overlay-mode option in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the containers retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve containers with any modification time stamp.

nameList

The names of the containers to retrieve.

containerTypes

A list of values that indicate the container types to retrieve.

`0:`	Retrieve all container types (`ARCON_ALL`).
`1:`	Retrieve all guide containers (`ARCON_GUIDE`).
`2:`	Retrieve all application containers (`ARCON_APP`).
`3:`	Retrieve all packing list containers (`ARCON_PACK`).
`4:`	Retrieve all filter guide containers (`ARCON_FILTER_GUIDE`).

attributes

Specify `AR_HIDDEN_INCREMENT` for this parameter to retrieve both visible and hidden containers. Specify `NULL` for this parameter to retrieve only visible containers.

ownerObjList

A list of schemas that own this container. This parameter can be `NULL` if the container exists globally.

refTypes

A list of the types of references (for example, forms and filters) to retrieve. You can specify individual types of references to retrieve, specify that all (`ARREF_ALL`) or none (`ARREF_NONE`) of the references be retrieved, or specify negative numbers to treat the types of references as exclude reference types. The exclude reference types take precedence over the include list; this means that if you specify a type as positive as well as negative, then all references of that type are excluded.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the containers exist. Values: `TRUE` means the container exists; `FALSE` means the container does not exist.

containerNameList

The array of container names retrieved.

assignedGroupListList

A list of zero or more groups that are directly assigned permission to the container.

groupListList

A list of zero or more groups who can access this container, including the inheritance hierarchy in the case of hierarchical groups. Access to this information is limited to users with BMC Remedy AR System administrator privileges. Specify `NULL` for this parameter if you do not want to retrieve this value.

If the static inheritance property is not enabled for the container, this list will be the same as the `assignedGroupListList`.

admingrpListList

A list of zero or more groups who can administer this container (and the referenced objects). If `ownerObj` does not return `NULL`, this list is the Subadministrator group list of the owning form. Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify `NULL` for this parameter if you do not want to retrieve the administrator group list.

ownerObjListList

A list of schemas that own this container. Specify `NULL` for this parameter if you do not want to retrieve this value. If this parameter returns `NULL`, the container exists globally.

labelList

A list of labels for this container. Specify `NULL` for this parameter if you do not want to retrieve the list.

descriptionList

The list of descriptions for this container. Specify `NULL` for this parameter if you do not want to retrieve the list.

typeList

The list of the container's type.

`0:`	Retrieve all container types (`ARCON_ALL`).
`1:`	Retrieve all guide containers (`ARCON_GUIDE`).
`2:`	Retrieve all application containers (`ARCON_APP`).
`3:`	Retrieve all packing list containers (`ARCON_PACK`).
`4:`	Retrieve all filter guide containers (`ARCON_FILTER_GUIDE`).

referenceList

The list of pointers to the objects (for example, forms or filters) referenced by this container. Specify `NULL` for this parameter if you do not want to retrieve the list.

helpTextList

A list of the help texts associated with the containers. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether instances of these objects exist).

ownerList

The list of owning users for the containers. Specify `NULL` for this parameter if you do not want to retrieve the list.

timestampList

The list of time stamps that specify the last change to the containers. Specify `NULL` for this parameter if you do not want to retrieve the list.

lastChangedList

The list of users who made the last change to the containers. Specify `NULL` for this parameter if you do not want to retrieve the list.

changeDiaryList

The list of change diary entries associated with the containers. The server adds the user making the change and a time stamp when it saves the change diary. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. Specify `NULL` for this parameter if you do not want to retrieve the change diary list. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropListList

A list of server object property lists that correspond to the containers in `nameList`. If this parameter is set to `NULL`, no properties are returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetContainer, ARGetListContainer.

6.2.92 ARGetMultipleCurrencyRatioSets

Description

Retrieves a list of formatted currency ratio sets valid for the times specified in the `ratioTimestamps` argument. You can use `ARGetCurrencyRatio` to extract a specific currency ratio from a ratio set that `ARGetMultipleCurrencyRatioSets` returns.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARGetMultipleCurrencyRatioSets(
   ARControlStruct *control,
   ARTimestampList *ratioTimestamps,
   ARTextStringList *currencyRatioSets,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

ratioTimestamps

A list of time stamps that represent times to retrieve sets of ratios. Specify the `AR_CURRENT_CURRENCY_RATIOS` constant as a time stamp to retrieve a set of current ratios for each conversion combination. If no ratio exists for a currency code combination exactly at the specified point in time, the closest earlier ratio is returned.

Return values

currencyRatioSets

A pointer to a structure that contains a list of formatted character strings, each of which represents the set of currency ratios for the corresponding point in time, as the `ratioTimestamps` argument specifies. If there are no available currency ratios that qualify, such as when there are no earlier ratios, the corresponding character string is an empty string.

You must free the memory allocated within the structure described previously. To determine the value for a specific ratio from the formatted character string, use the function `ARGetCurrencyRatio`.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetCurrencyRatio. See FreeAR for: `FreeARTextStringList`.

6.2.93 ARGetMultipleEntryPoints

Description

Retrieves the entry points of multiple applications. It returns the entry points that are accessible by the user, taking into account user permissions, licenses, and application states.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleEntryPoints(
   ARControlStruct *control,
   ARTimeStamp changedSince,
   ARNameList *appNameList,
   ARReferenceTypeList *refTypeList,
   ARNameType *displayTag,
   unsigned int *vuiType,
   ARNameList *entryPointNameList,
   ARUnsignedIntList *entryPointTypeList,
   ARTextStringList *entryPointDLabelList,
   ARNameList *ownerAppNameList,
   ARTextStringList *ownerAppDLabelList
   ARPermissionListList *groupListList,
   ARContainerOwnerObjListList *ownerObjListList,
   ARTextStringList *descriptionList,
   ARReferenceListList *referencesList,
   ARTextStringList *helpTextList,
   ARTimestampList *timestampList,
   ARPropListList *objPropListList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

changedSince

A time stamp that limits the objects retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve objects with any modification time stamp.

appNameList

A list of applications objects that contains the entry points.

refTypeList

A list of reference types to be retrieved.

displayTag

The name of the form view (VUI) to use for the entry point label. If the specified view does not exist or you specify `NULL` for this parameter, the system uses the default view.

vuiType

The type of VUI to retrieve. Specify `NULL` for this parameter if you do not want to retrieve the VUI name.

`0:`	No VUI type (`AR_VUI_TYPE_NONE`).
`1:`	Windows type (`AR_VUI_TYPE_WINDOWS`)* — fields in BMC Remedy User.
`2:`	Web relative type (`AR_VUI_TYPE_WEB`) — fields in view can be adjusted.
`3:`	Web absolute type (`AR_VUI_TYPE_WEB_ABS_POS`) — fields in view are fixed.

Fields marked with a * are available for legacy environments that use BMC Remedy User, which is no longer shipped with BMC Remedy AR System.

Return values

entryPointNameList

A list of entry point names. If the entry point type is an active link guide, this is the name of the container. If the entry point type is a form, this is the name of the form.

entryPointTypeList

A list of entry point types.

`1:`	The entry point type for an active link guide (`AR_ENTRYPOINT_TYPE_AL_GUIDE`).
`2:`	The entry point type for a form in default search mode (`AR_ENTRYPOINT_TYPE_DEFAULT_SEARCH`).
`3:`	The entry point type for a form in default new entry mode (`AR_ENTRYPOINT_TYPE_DEFAULT_NEW`).

entryPointDLabelList

Display label for entry point, which is optionally localized.

ownerAppNameList

The application owner of the entry point.

ownerAppDLabelList

The display label for the application owner, which is optionally localized. This is the localized label for the application container object.

groupListList

A list of zero or more groups who can access this form or active link guide. Access to this information is limited to users with BMC Remedy AR System administrator privileges. Specify `NULL` for this parameter if you do not want to retrieve the group list.

ownerObjListList

For active link guide entry points, a list of the schemas that own this container. For form entry points, this is the name of the form that the default entry point came from. Specify `NULL` for this parameter if you do not want to retrieve this value. If this parameter returns `NULL`, the container exists globally.

descriptionList

For active link guide entry points, this is the list of descriptions for this entry point. Specify `NULL` for this parameter if you do not want to retrieve the list.

referencesList

A list of pointers to the objects (for example, forms or filters) referenced by this container. Specify `NULL` for this parameter if you do not want to retrieve this value.

helpTextList

A list of the help texts associated with the entry point. Specify `NULL` for this parameter if you do not want to retrieve the help texts (which is useful if you are calling this function to verify whether instances of these objects exist).

timestampList

A list of time stamps that specify the last change to the entry point object. Specify `NULL` for this argument if you do not want to retrieve this value.

objPropListList

A list of server object property lists. If this parameter is set to `NULL`, no properties are returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetMultipleContainers.

6.2.94 ARGetMultipleEntries

Description

Retrieves the group of form entries specified by the `entryIdList` parameter. You can retrieve data for specific fields or all (accessible) fields. This function returns only the name, size, and compressed size of attachment fields. Use `ARGetEntryBLOB` to retrieve the contents of the attachment. This function performs the same action as `ARGetEntry` but is easier to use and more efficient than retrieving multiple entries one by one.

Note

A maximum of 100 entries can be returned by this function. If you need to return more than 100 entries, call this function multiple times.

Privileges

The system returns data based on the access privileges of the user that you specify for the `control` parameter. User permissions are verified for each specified entry and field. The system returns values for accessible fields and warning messages for fields the user cannot access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleEntries(
   ARControlStruct *control,
   ARNameType schema,
   AREntryIdListList *entryIdList,
   ARInternalIdList *IdList,
   ARBooleanList *existList,
   ARFieldValueListList *fieldList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form containing the entries to retrieve.

entryIdList

The list of zero or more entries to retrieve.

IdList

A list of zero or more internal IDs that specify the fields to retrieve. Specify `NULL` for this parameter (or zero fields) to retrieve all (accessible) fields. To improve performance, specify only the fields that you need if you do not require the data for all fields. If an attachment field is specified in the list, only its name, size, and compressed size are returned. Use `ARGetEntryBLOB` to retrieve the contents of the attachment.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the entries exist. Values: `TRUE` means the entry exists; `FALSE` means the entry does not exist.

fieldList

A list of zero or more fields and associated values that identify the data for the specified entries. The fields are returned in the order specified by `fieldIdList`. If a specified entry does not exist, the system returns zero (or `NULL`) for that entry. If the user does not have permission for a specified field or the field does not exist, the system returns `NULL` for the associated value within each entry value list.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEntry, ARDecodeDiary, ARDecodeStatusHistory, ARDeleteEntry, ARGetEntry, ARGetListEntry, ARGetMultipleActiveLinks, ARGetMultipleExtFieldCandidates, ARGetMultipleFields, ARGetMultipleLocalizedValues, ARGetMultipleSchemas, ARSetEntry. See FreeAR for: `FreeARBooleanList`, `FreeAREntryIdList`, `FreeARInternalIdList`, `FreeARStatusList`.

6.2.95 ARGetMultipleEscalations

Description

Retrieves from the specified server information about the escalations whose names are specified by the `nameList` parameter. This function performs the same action as `ARGetEscalation` but is easier to use and more efficient than retrieving information about multiple escalations one by one. Information is returned in lists for each item, with one item in the list for each escalation returned. For example, if the second item in the list for `existList` is `TRUE`, the time stamp that specifies the last change to the second escalation is returned in the second item in the list for `timeStampList`.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleEscalations(
   ARControlStruct *control,
   ARTimestamp changedSince,
   ARNameList *nameList,
   ARBooleanList *existList,
   ARNameList *esclNameList,
   AREscalationTmList *esclTmList,
   ARWorkflowConnectList *workflowConnectList,
   ARUnsignedIntList *enableList,
   ARQualifierList *queryList,
   ARFilterActionListList *actionListList,
   ARFilterActionListList *elseListList,
   ARTextStringList *helpTextList,
   ARTimestampList *timestampList,
   ARAccessNameList *ownerList,
   ARAccessNameList *lastChangedList,
   ARTextStringList *changeDiaryList,
   ARPropListList *objPropListList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the escalations retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve escalations with any modification time stamp.

nameList

The names of the escalations to retrieve. The nameList can be passed as a `NULL` or as an empty list, as in:

```
ARNameList emptyNameList = {0, 0};
   ...
   ARGetMultipleEscalations(control, changedSince, &emptyNameList, ...)
```

In this case information is returned for every escalation that passes the `changedSince` condition.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the escalations exist. Values: `TRUE` means the escalation exists; `FALSE` {means the escalation does not exist.

esclNameList

A list of zero or more escalations that match the criteria defined in the `changedSince` and `nameList` parameters. The system returns a list with zero items if no escalations match the specified criteria.

esclTMList

A list of `AREscalationTmStruct`. `ARGetEscalation` returns a single `AREscalationTmStruct`, which describes when the escalation executes, such as once an hour or on specified days.

workflowConnectList

A list of the items that ARGetEscalation returns one at a time, which are the schemas to which the escalation is attached.

enableList

A list of flags that specify whether the escalation is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve these flags.

queryList

A list of qualifications that determines whether the escalations are executed. The system returns zero (`AR_COND_OP_NONE`) if the escalations have no qualifications. Specify `NULL` for this parameter if you do not want to retrieve the queries.

actionListList

A list of the sets of actions performed if the condition defined by the `query` parameter is satisfied. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve the action lists.

elseListList

A list of the sets of actions performed if the conditions defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve the else list.

helpTextList

A list of help text items associated with the escalations. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of these objects exists).

timeStampList

A list of time stamps that specify the last change to the escalations. Specify `NULL` for this parameter if you do not want to retrieve this value.

ownerList

A list of owning users for the escalations. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChangedList

A list of users who made the last change to the escalations. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiaryList

A list of change diaries associated with the escalations. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary list. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropListList

A list of server object property lists that correspond to the escalations in `nameList`. If this parameter is set to `NULL`, no properties are returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEscalation, ARDecodeDiary, ARDeleteEscalation, ARGetEscalation, ARGetListEscalation, ARSetEscalation. See FreeAR for: `FreeARFilterActionList`, `FreeARPropList`, `FreeARQualifierStruct`, `FreeARStatusList`.

6.2.96 ARGetMultipleExtFieldCandidates

Description

Retrieves a list of all available external data source fields (field candidates). Users choose the data from one of these field candidates to populate the vendor form.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to fields the user can access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleExtFieldCandidates(
   ARControlStruct *control,
   ARCompoundSchema *schema,
   ARFieldMappingList *fieldMapping,
   ARFieldLimitList *limit,
   ARUnsignedIntList *dataType,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The definition of the candidate form or table that contains the candidate field to retrieve.

Return values

fieldMapping

A list of candidate field definitions.

limit

A list of field limits for the candidate fields.

datatype

A list of field datatypes for the candidate fields.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListExtSchemaCandidates, ARGetMultipleActiveLinks, ARGetMultipleEntries, ARGetMultipleFields, ARGetMultipleLocalizedValues, ARGetMultipleSchemas. See FreeAR for: `FreeARCompoundSchema`, `FreeARFieldMappingList`, `FreeARFieldLimitList`, `FreeARUnsignedIntList`, `FreeARStatusList`.

6.2.97 ARGetMultipleFields

Description

Retrieves one or more field definitions for a specified form. This function performs the same action as `ARGetField` but is easier to use and more efficient than retrieving information about multiple fields one by one.

Information is returned in lists for each item, with one item in the list for each field returned. For example, if the second item in the list for `existList` is `TRUE`, the name of the second field is returned in the second item in the list for `fieldName`.

Privileges

This operation can be performed by users with any level of access permission for the fields in the specified form. Access to `permissions` information is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleFields(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalIdList *fieldId,
   ARBooleanList *existList,
   ARInternalIdList *fieldId2,
   ARNameList *fieldName,
   ARFieldMappingList *fieldMap,
   ARUnsignedIntList *dataType,
   ARUnsignedIntList *option,
   ARUnsignedIntList *createMode,
   ARUnsignedIntList *fieldOption,
   ARValueList *defaultVal,
   ARPermissionListList *assignedGroupListList,
   ARPermissionListList *permissions,
   ARFieldLimitList *limit,
   ARDisplayInstanceListList *dInstanceList,
   ARTextStringList *helpText,
   ARTimestampList *timestamp,
   ARAccessNameList *owner,
   ARAccessNameList *lastChanged,
   ARTextStringList *changeDiary,
   ARPropListList *objPropListList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for `Overlay-mode` in ar.cfg or ar.conf options N-R.

schema

The name of the form containing the field to retrieve.

fieldId

The internal ID list of fields to retrieve. Specify `NULL` or `fieldIdList.numItems=0` for this parameter to retrieve data for *all* the fields in the form.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the fields exist. Values: `TRUE` means the field exists; `FALSE` means the field does not exist.

fieldId2

The list of internal IDs of the fields retrieved. Specify `NULL` for this parameter if you do not want to retrieve the field IDs.

fieldName

The list of names of the fields. Specify `NULL` for this parameter if you do not want to retrieve the field names.

fieldMap

A mapping to the underlying form from which to retrieve the field. Useful for join forms. Specify `NULL` for this parameter if you do not want to retrieve the field mapping. See Mapping fields in schemas.

dataType

The list data types of the fields. See ARCreateField for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve the data types.

option

A list of flags that indicate whether users must enter a value in the fields. See ARCreateField for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this flag.

createMode

A list of flags that indicate the permission status for the fields when users submit entries. See ARCreateField for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this value.

fieldOption

A list of bitmasks that indicates whether the field is to be audited or copied when other fields are audited.

Bit 0:	Audit this field. (`AR_FIELD_BITOPTION_AUDIT`)
Bit 1:	Copy this field when other fields in the form are audited. (`AR_FIELD_BITOPTION_COPY`)
Bit 2:	Indicates this field is for Log Key 1. (`AR_FIELD_BITOPTION_LOG_KEY1`)
Bit 3:	Indicates this field is for Log Key 2. (`AR_FIELD_BITOPTION_LOG_KEY2`)
Bits 2 and 3:	Indicates this field is for Log Key 3. (`AR_FIELD_BITOPTION_LOG_KEY3`)

defaultVal

The list of values to apply if a user submits an entry with no field value (only applicable to data fields). The system returns `0` (`AR_DEFAULT_VALUE_NONE`) if the field has no default. Specify `NULL` for this parameter if you do not want to retrieve the default values.

assignedGroupListList

A list of zero or more groups that are directly assigned permission to the field.

permissions

A list of lists containing zero or more groups who can access the fields retrieved, including the inheritance hierarchy in the case of hierarchical groups. Access to this information is limited to users with BMC Remedy AR System administrator privileges. Specify `NULL` for this parameter if you do not want to retrieve the permissions.

If the static inheritance property is not enabled for the field, this list will be the same as the `assignedGroupListList`.

limit

The list of value limits for the fields and other properties specific to the individual fields' respective types. See Defining field limits for a description of the contained structure that applies to the type of field that you are creating. Specify `NULL` for this parameter if you do not want to retrieve the limits and properties.

dInstanceList

A list of lists containing zero or more display properties associated with the fields. See ARCreateField for a description of the possible values. The system returns `0` (`AR_DPROP_NONE`) if the field has no display properties. Specify `NULL` for this parameter if you do not want to retrieve this list.

helpText

The list of help text associated with the fields. Specify `NULL` for this parameter if you do not want to retrieve the help text.

timestamp

A list of time stamps that specify the last change to the fields. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The list of owning users for the field. Specify `NULL` for this parameter if you do not want to retrieve the owner.

lastChanged

The list of users who made the last change to the fields. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiary

The list of change diary entries associated with the fields. The server adds the user making the change and a time stamp when it saves the change diary. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. Specify `NULL` for this parameter if you do not want to retrieve the change diary.

objPropListList

A list of server object property lists that correspond to the fields in `fieldId`. If this parameter is set to `NULL`, no properties are returned. See Server object properties and structures.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateField, ARDecodeDiary, ARDeleteField, ARDeleteMultipleFields, ARGetField, ARGetListField, ARGetMultipleActiveLinks, ARGetMultipleEntries, ARGetMultipleExtFieldCandidates, ARGetMultipleLocalizedValues, ARGetMultipleSchemas, ARGetSchema, ARSetField. See FreeAR for: `FreeARBooleanList`, `FreeARAccessNameList`, `FreeARDisplayInstanceListList`, `FreeARFieldLimitList`, `FreeARFieldMappingList`, `FreeARInternalIdList`, `FreeARNameList`, `FreeARPermissionListList`, `FreeARStatusList`, `FreeARTextStringList`, `FreeARTimestampList`, `FreeARUnsignedIntList`, `FreeARValueList`.

6.2.98 ARGetMultipleFilters

Description

Retrieves from the specified server information about the filters whose names are specified by the `nameList` parameter. This function performs the same action as `ARGetFilter` but is easier to use and more efficient than retrieving information about multiple filters one by one. Information is returned in lists for each item, with one item in the list for each filter returned. For example, if the second item in the list for `existList` is `TRUE`, the name of the second filter is returned in the second item in the list for `filterNameList`.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleFilters(
   ARControlStruct *control,
   ARTimestamp changedSince,
   ARNameList *nameList,
   ARBooleanList *existList,
   ARNameList *filterNameList,
   ARUnsignedIntList *orderList,
   ARWorkflowConnectList *workflowConnectList,
   ARUnsignedIntList *opSetList,
   ARUnsignedIntList *enableList,
   ARQualifierList *queryList,
   ARFilterActionListList *actionListList,
   ARFilterActionListList *elseListList,
   ARTextStringList *helpTextList,
   ARTimestampList *timestampList,
   ARAccessNameList *ownerList,
   ARAccessNameList *lastChangedList,
   ARTextStringList *changeDiaryList,
   ARPropListList *objPropListList,
   ARUnsignedIntList *errorFilterOptionsList,
   ARNameList *errorFilterNameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the filters retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve filters with any modification time stamp.

nameList

The names of the filters to retrieve. The nameList can be passed as a `NULL` or as an empty list, as in:

```
ARNameList emptyNameList = {0, 0};
   ...
   ARGetMultipleFilters(control, changedSince, &emptyNameList, ...)
```

In this case information is returned for every filter that passes the `changedSince` condition.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the filters exist. Values: `TRUE` means the filter exists; `FALSE` means the filter does not exist.

filterNameList

A list of zero or more filters that match the criteria defined in the `changedSince` and `nameList` parameters. The system returns a list with zero items if no filters match the specified criteria.

orderList

A list of values between `0` and `1000` (inclusive) that determines the filter execution order. When multiple filters are associated with a form, the value associated with each filter determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnectList

A list of the items that `ARGetFilter` returns one at a time, which are the schemas to which the filter is attached.

opSetList

Each member of `opSetList` is an integer bit mask that indicate on which conditions the filter executes: form get, create, delete, update, and merge. For more information, see ARCreateFilter.

enableList

A list of flags that specify whether the filter is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve these flags.

queryList

A list of qualifications that determines whether the filters are executed. The system returns zero (`AR_COND_OP_NONE`) if the filters have no qualifications. Specify `NULL` for this parameter if you do not want to retrieve the queries.

actionListList

A list of the sets of actions performed if the condition defined by the `query` parameter is satisfied. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve the action lists.

elseListList

A list of the sets of actions performed if the conditions defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve the else list.

helpTextList

A list of help text items associated with the filters. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of these objects exists).

timeStampList

A list of time stamps that specify the last change to the filters. Specify `NULL` for this parameter if you do not want to retrieve this value.

ownerList

A list of owning users for the filters. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChangedList

A list of users who made the last change to the filters. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiaryList

A list of change diaries associated with the filters. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary list. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropListList

A list of object property lists that correspond to the filters in `nameList`. If this parameter is set to `NULL`, no properties are returned. See Server object properties.

errorFilterOptionsList

A list of error handler option bitmaps. An item in the list is `AR_FILTER_ERRHANDLER_ENABLE` if the error handler filter is enabled, zero if not. Specify `NULL` for this parameter if you do not want to retrieve this value.

errorFilterNameList

A list of names of error handler filters. An item in the list is `NULL` if no error handler is specified for the filter. Specify `NULL` for this parameter if you do not want to retrieve this value.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateFilter, ARDecodeDiary, ARDeleteFilter, ARGetFilter, ARGetListFilter, ARSetFilter. See FreeAR for: `FreeARFilterActionList`, `FreeARPropList`, `FreeARQualifierStruct`, `FreeARStatusList`.

6.2.99 ARGetMultipleImages

Description

Retrieves information from the specified server about the images whose names are specified in the `nameList` parameter. This function performs the same action as `ARGetImage`, but it is more efficient than retrieving information about multiple images one by one. Information is returned in lists for each item, with one item in the list for each image returned. For example, if the second item in the list for `existList` is TRUE, the name of the second image is returned in the second item in the list for `imageNameList`.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleImages(
   ARControlStruct *control,
   ARTimestamp changedSince,
   ARNameList *nameList,
   ARBooleanList *existList,
   ARNameList *imageNameList,
   ARTextStringList *imageTypeList,
   ARFilterActionListList *elseListList,
   ARTimestampList *timestampList,
   ARTextStringList *descriptionList,
   ARTextStringList *helpTestList,
   ARAccessNameList *ownerList,
   ARAccessNameList *lastChangedList,
   ARTextStringList *changeDiaryList,
   ARPropListList *objPropListList,
   ARTextStringList *checkSumList,
   ARImageDataList *imageDataList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the images retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve images with any modification time stamp.

nameList

The names of the images to retrieve. The `nameList` can be passed as a `NULL` or as an empty list, as in:

```
ARNameList emptyNameList = {0, 0};
   ...
   ARGetMultipleImages(control, changedSince, &emptyNameList, ...)
```

In this case information is returned for every image that passes the `changedSince` condition.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the images exist. Values: `TRUE` means the image exists; `FALSE` means the image does not exist.

imageNameList

A list of zero or more images that match the criteria defined in the `changedSince` and `nameList` parameters. The value is a list with zero items if no images match the specified criteria.

imageTypeList

A list of strings giving the type of the corresponding image in `imageNameList`. Specify `NULL` for this parameter if you do not want to retrieve this value.

timestampList

A list of time stamps that specify the last change to the corresponding image in `imageNameList`. Specify `NULL` for this parameter if you do not want to retrieve this value.

descriptionList

A list of descriptions of the corresponding image in `imageNameList`. Specify `NULL` for this parameter if you do not want to retrieve this value.

helpTextList

A list of help texts for the corresponding image in `imageNameList`. Specify `NULL` for this parameter if you do not want to retrieve this value. This value can be large, so do not reactive it if the program does not need it, for example, if the program is only checking it an object exists.

ownerList

A list of owning users for the corresponding image in `imageNameList`. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChangedList

A list of users who made the last change to the corresponding image in `imageNameList`. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiaryList

A list of change diaries associated with the corresponding image in `imageNameList`. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary list. This value can be large, so do not reactive it if the program does not need it, for example, if the program is only checking it an object exists.

objPropListList

A list of server object property lists that correspond to the images in `imageNameList`. If this parameter is set to `NULL`, no properties are returned. See Server object properties and structures.

checkSumList

A list of MD5 hash values retrieved from the database for the corresponding image data in `imageDataList` . Your program can compute an MD5 has value and check it against this value to check whether the image data was retrieved correctly. Specify `NULL` for this parameter if you do not want to retrieve this value.

imageDataList

An array with the data for the corresponding image in `imageNameList`. See Images and structures. Specify `NULL` for this parameter if you do not want to retrieve this value.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateImage, ARDeleteImage, ARGetImage, ARGetListImage, ARSetImage. See FreeAR for: `FreeARImageDataStruct`, `FreeARStatusList`.

6.2.100 ARGetMultipleLocalizedValues

Description

Retrieves multiple localized text strings from the BMC Remedy Message Catalog. The messages that the server retrieves depend on the user locale in the `control` structure. This function performs the same action as `ARGetLocalizedValues` but is easier to use and more efficient than retrieving multiple values one by one.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleLocalizedValues(
   ARControlStruct *control,
   ARLocalizedRequestList *localizedRequestList,
   ARValueList *localizedValueList,
   ARTimestampList  *timestampList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

localizedRequestList

A list of identifications for the localized value to be retrieved. The list contains the type of value, value name, and, for some types of values, unique identification numbers.

Return values

localizedValueList

A list of the localized values to be retrieved. The values can be a character value or an attachment. Specify `NULL` for this argument if you do not want to retrieve a value.

timestampList

A list of time stamps that specify the last change to the values. Specify `NULL` for this argument if you do not want to retrieve this value.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetLocalizedValue, ARGetMultipleActiveLinks, ARGetMultipleEntries, ARGetMultipleExtFieldCandidates, ARGetMultipleFields, ARGetMultipleSchemas. See FreeAR for: `FreeARLocalizedRequestList`, `FreeARValueList`, `FreeARTimestampList`, `FreeARStatusList`.

6.2.101 ARGetMultipleSchemas

Description

Retrieves information about multiple forms with the indicated names from the specified server. This information does not include the form field definitions (see ARGetField or ARGetMultipleFields). This function performs the same action as `ARGetSchema` but is easier to use and more efficient than retrieving information about multiple forms one by one.

Information is returned in lists for each item, with one item in the list for each form returned. For example, if the second item in the list for `existList` is `TRUE`, the name of the second form is returned in the second item in the list for `schemaNameList`.

Privileges

This operation can be performed by users with access permission for the form. Access to `groupList` and `admingrpList` information is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleSchemas(
   ARControlStruct *control,
   ARTimestamp changedSince,
   ARUnsignedIntList *schemaTypeList,
   ARNameList  *nameList,
   ARInternalIdList  *fieldIdList,
   ARBooleanList *existList,
   ARNameList *schemaNameList,
   ARCompoundSchemaList  *schemaList,
   ARSchemaInheritanceListList *schemaInheritanceListList,
   ARPermissionListList *assignedGroupListList,
   ARPermissionListList  *groupListList,
   ARInternalIdListList  *admingrpListList,
   AREntryListFieldListList  *getListFieldsList,
   ARSortListList  *sortListList,
   ARIndexListList  *indexListList,
   ARArchiveInfoList archiveInfoList.
   ARAuditInfoList auditInfoList.
   ARNameList *defaultVuiList,
   ARTextStringList  *helpTextList,
   ARTimestampList  *timestampList,
   ARAccessNameList  *ownerList,
   ARAccessNameList  *lastChangedList,
   ARTextStringList  *changeDiaryList,
   ARPropListList  *objPropListList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

changedSince

A time stamp that limits the forms retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve forms with any modification time stamp.

schemaTypeList

A list of values that indicate the form types to retrieve (some types are mutually exclusive).

Bit 0:	Retrieve a list of all form types (`AR_LIST_SCHEMA_ALL`).

Bit 1:	Retrieve a list of all base forms (AR_LIST_SCHEMA_REGULAR).
Bit 2:	Retrieve a list of all join forms (AR_LIST_SCHEMA_JOIN).
Bit 3:	Retrieve a list of all view forms (AR_LIST_SCHEMA_VIEW).
Bit 6:	Retrieve a list of all display-only forms (AR_LIST_SCHEMA_DIALOG).
Bit 7:	Retrieve a list of all forms with data fields (AR_LIST_SCHEMA_ALL_WITH_DATA).
Bit 8:	Retrieve a list of all vendor schemas (AR_SCHEMA_VENDOR).

To retrieve both visible and hidden forms, add `1024` (`AR_HIDDEN_INCREMENT`) to the form type. For example, specify `AR_LIST_SCHEMA_ALL | AR_HIDDEN_INCREMENT` for this parameter. This function does not support the `AR_LIST_SCHEMA_UPLINK` and `AR_LIST_SCHEMA_DOWNLINK` form types.

nameList

The names of the schemas to retrieve.

fieldIdList

List of form fields that exist in the retrieved schema. The system returns only the forms that contain all the fields in this list. Specify `NULL` for this parameter (or zero fields) to retrieve all (accessible) fields.

Note

Archive forms are not returned if you specify a list of form fields. You must specify `NULL` to return archive forms.

archiveInfoList

The list of archive information associated with the form. Specify `NULL` for this parameter to not retrieve archive information.

auditInfoList

The list of audit information associated with the form. Specify `NULL` for this parameter to not retrieve audit information.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the forms exist. Values: TRUE means the forms exists; FALSE means the form does not exist.

schemaNameList

An array of retrieved schema names.

schemaList

A list of form types (base form or join form). See ARCreateSchema for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this value.

schemaInheritanceListList

This is reserved for future use. Specify `NULL`.

assignedGroupListList

A list of zero or more groups that are directly assigned permission to the forms.

groupListList

A list of zero or more groups who can access the forms, including the inheritance hierarchy in the case of hierarchical groups. Access to this information is limited to users with BMC Remedy AR System administrator privileges. Specify `NULL` for this parameter if you do not want to retrieve the group list.

If the static inheritance property is not enabled for the form, this list will be the same as the `assignedGroupListList`.

admingrpListList

A list of zero or more groups who can administer this form (and the associated filters, escalations, and active links). Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify `NULL` for this parameter if you do not want to retrieve the administrator group list.

getListFieldsList

A list of zero or more fields that identifies the default query list data for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve the default query list fields.

sortListList

A list of zero or more fields that identifies the default sort order for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve this value.

indexListList

The set of zero or more indexes for the form. Specify `NULL` for this parameter if you do not want to retrieve the index list.

defaultVuiList

The default VUI label. Specify `NULL` for this parameter if you do not want to retrieve the index list.

helpTextList

A list of the help texts associated with the forms. Specify `NULL` for this parameter if you do not want to retrieve the help texts (which is useful if you are calling this function to verify whether instances of these objects exist).

timestampList

A list of time stamps that identify the last changes to the forms. Specify `NULL` for this parameter if you do not want to retrieve this value.

ownerList

A list of form owners. Specify `NULL` for this parameter if you do not want to retrieve the owners.

lastChangedList

A list of users who made the last changes to the forms. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiaryList

A list of the change diaries associated with the forms. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropListList

A list of server object property lists that correspond to the schemas in `schemaNameList`. If this parameter is set to `NULL`, a properties list with zero properties is associated with the object, and when an `ARGetSchema` action is performed, zero properties are returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateSchema, ARDeleteSchema, ARGetSchema, ARGetListSchema, ARGetMultipleActiveLinks, ARGetMultipleEntries, ARGetMultipleExtFieldCandidates, ARGetMultipleFields, ARGetMultipleLocalizedValues, ARSetSchema. See FreeAR for: `FreeARAccessNameList`, `FreeARBooleanList`, `FreeARCompoundSchemaList`, `FreeAREntryListFieldListList`, `FreeARIndexListList`, `FreeARInternalIdList`, `FreeARInternalIdListList`, `FreeARNameList`, `FreeARPermissionListList`, `FreeARPropListList`, `FreeARSortListList`, `FreeARStatusList`, `FreeARTextStringList`, `FreeARTimestampList`, `FreeARUnsignedIntList`.

6.2.102 ARGetMultipleVUIs

Description

Retrieves from the specified server information about the form views (VUIs) whose names are specified by the `nameList` parameter. This function performs the same action as `ARGetVUI` but is easier to use and more efficient than retrieving information about multiple form views one by one. Information is returned in lists for each item, with one item in the list for each VUI returned. For example, if the second item in the list for `existList` is TRUE, the name of the second VUI is returned in the second item in the list for `nameList`.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetMultipleVUIs(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalIdList *wantList,
   ARTimestamp changedSince,
   ARBooleanList *existList,
   ARInternalIdList *gotList,
   ARNameList *nameList,
   ARLocalList *localeList,
   ARUnsignedIntList *vuiTypeList,
   ARPropListList *dPropListList,
   ARTextStringList *helpTextList,
   ARTimestampList *timestampList,
   ARAccessNameList *ownerList,
   ARAccessNameList *lastChangedList,
   ARTextStringList *changeDiaryList,
   ARPropList *objPropListList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

schema

The names of the forms containing the VUIs to retrieve.

wantList

The IDs of the VUIs to retrieve. The `wantList` can be passed as a `NULL` or as an empty list, as in:

```
ARWantList emptyWantList = {0, 0};
   ...
   ARGetMultipleVUIs(control, changedSince, &emptyWantList, ...)
```

In this case information is returned for every VUI that passes the `changedSince` condition.

changedSince

A time stamp that limits the VUIs retrieved to those modified after the specified time. Specify `0` for this parameter to retrieve VUIs with any modification time stamp.

Return values

existList

A list of flags and corresponding Boolean values that indicate whether the VUIs exist and meet the qualifying criteria. Values: `TRUE` means the VUI exists; `FALSE` means the VUI does not exist.

nameList

A list of VUIs retrieved.

localeList

A list of locales retrieved.

vuiTypeList

A list of VUI types retrieved.

dPropListList

A list of display properties of VUIs. Specify `NULL` for this parameter if you do not want to retrieve this value.

helpTextList

A list of help text items associated with the VUIs. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of these objects exists).

timeStampList

A list of time stamps that specify the last change to the VUIs. Specify `NULL` for this parameter if you do not want to retrieve this value.

ownerList

A list of owning users for the VUIs. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastChangedList

A list of users who made the last change to the VUIs. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiaryList

A list of change diaries associated with the VUIs. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary list. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropListList

A list of object property lists that correspond to the VUIs in `nameList`. If this parameter is set to `NULL`, no properties are returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateVUI, ARDecodeDiary, ARDeleteVUI, ARGetListVUI, ARGetVUI, ARSetVUI. See FreeAR for: `FreeARPropList`, `FreeARStatusList`.

6.2.103 ARGetObjectChangeTimes

Description

Retrieves timestamps for the last create, modify, and delete operations for each type of server object.

Privileges

All users.

Synopsis

```
#include "ar.h"

int ARGetObjectChangeTimes(
   ARControlStruct *control,
   ARObjectChangeTimestampList *objectChanges,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

ARObjectChangeTimestampList

A list of `ARObjectChangeTimestamps`. Each item in this structure contains an object type and three timestamps; the timestamps are the last time an object of the specified type was created, changed, and deleted.

All timestamps are effective as of when the server was started. If the server is restarted all timestamps will be zero.

The object types are:

- `AR_STRUCT_ITEM_SCHEMA`
- `AR_STRUCT_ITEM_FILTE`
- `AR_STRUCT_ITEM_ACTIVE_LIN`
- `AR_STRUCT_ITEM_CHAR_MEN`
- `AR_STRUCT_ITEM_ESCALATIO`
- `AR_STRUCT_ITEM_CONTAINE`
- `AR_STRUCT_ITEM_IMAGE`

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.104 ARGetOneEntryWithFields

Description

Retrieves one entry from BMC Remedy AR System matching a given qualification. Similar to `ARGetListEntryWithFields`, with the following behavioral changes:

- If the qualifier in the API call matches multiple records, only the first is returned.
- Get filters on the form being queried are fired for that one record.

This function is equivalent to issuing an `ARGetListEntry` call followed by `ARGetEntry` with one of the entry IDs.

The keyword `$LASTCOUNT$` is set to the number of entries matched, even though only one is returned.

Privileges

The system returns information based on the access privileges of the user that you specify for the `control` parameter. All lists, therefore, are limited to entries the user can access (users must have permission for the `entryId` field to access and retrieve entries).

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetOneEntryWithFields(
   ARControlStruct *control,
   ARNameType schema,
   ARQualifierStruct *qualifier,
   AREntryListFieldList *getListFields,
   ARSortList *sortList,
   ARBoolean useLocale,
   AREntryListFieldValueList *entryList,
   unsigned int *numMatches,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The schema of which the entry is a member.

qualifier

Qualification to limit entries retrieved.

getListFields

Fields returned for matching entries.

sortList

The sort order for returned data.

useLocale

A flag that indicates whether to search for entries based on the locale. If you specify `1` (`TRUE`) and the Localize Server option is selected, entries are searched using the locale specified in `AR_RESERV_LOCALE_LOCALIZED_SCHEMA` (field ID 160)**.** If no matches are found for the specified locale, the search becomes less restrictive until a match is found. If you specify `0` (`FALSE`) or the **Localize Server** option is cleared, all entries are searched. For information, see Setting the Localize Server option.

Return values

entryList

A list of zero or more matching entries.

numMatches

The number of matches for qualifier, greater than or equal to the number returned in entryList.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListEntryWithFields

6.2.105 ARGetSchema

Description

Retrieves information about the form with the indicated name from the specified server. This information does not include the form's field definitions (see ARGetField).

Privileges

This operation can be performed by users with access permission for the form. Access to `groupList` and `admingrpList` information is limited to users with BMC Remedy AR System administrator privileges.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetSchema(
   ARControlStruct *control,
   ARNameType name,
   ARCompoundSchema *schema,
   ARSchemaInheritanceList *schemaInheritanceList,
   ARPermissionList *assignedGroupList,
   ARPermissionList *groupList,
   ARInternalIdList *admingrpList,
   AREntryListFieldList *getListFields,
   ARSortList *sortList,
   ARIndexList *indexList,
   ARArchiveInfoStruct *archiveInfo,
   ARAuditInfoStruct *auditInfo,
   ARNameType defaultVui,
   char **helpText,
   ARTimestamp *timestamp,
   ARAccessNameType owner,
   ARAccessNameType lastChanged,
   char **changeDiary,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the AR_SESS_CONTROL_PROP_API_OVERLAYGROUP variable of the ARSetSessionConfiguration function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the AR_SESS_CONTROL_PROP_API_OVERLAYGROUP variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for Overlay-mode in ar.cfg or ar.conf options N-R.

name

The name of the form to retrieve.

Return values

schema

The form type (base form or join form). See ARCreateSchema for a description of the possible values. Specify NULL for this parameter if you do not want to retrieve the form type.

schemaInheritance

This is reserved for future use. Specify NULL.

assignedGroupList

A list of zero or more groups that are directly assigned permission to the form.

groupList

A list of zero or more groups who can access the form, including the inheritance hierarchy in the case of hierarchical groups. Access to this information is limited to users with BMC Remedy AR System administrator privileges. Specify `NULL` for this parameter if you do not want to retrieve the group list.

If the static inheritance property is not enabled for the form, this list will be the same as the `assignedGroupList`.

admingrpList

A list of zero or more groups who can administer this form (and the associated filters, escalations, and active links). Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify `NULL` for this parameter if you do not want to retrieve the administrator group list.

getListFields

A list of zero or more fields that identifies the default query list data for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve this value.

sortList

A list of zero or more fields that identifies the default sort order for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve the default sort fields.

indexList

The set of zero or more indexes for the form. Specify `NULL` for this parameter if you do not want to retrieve the index list.

archiveInfo

If a form is to be archived, this is the archive information for the form. Specify `NULL` for this parameter if you do not want to retrieve the archive information. These are the values for archive type:

`0:`	Deletes the archive settings for the selected form. The selected form is not archived (`AR_ARCHIVE_NONE`).
`1:`	Entries on the selected form are copied to the archive form (`AR_ARCHIVE_FORM`).
`2:`	Entries on the selected form are deleted from the source form (`AR_ARCHIVE_DELETE`).
`4:`	Entries on the selected form are copied to an XML format file (`AR_ARCHIVE_FILE_XML`).
`8:`	Entries on the selected form are copied to an ARX format file (`AR_ARCHIVE_FILE_ARX`).
`32:`	Attachment fields are not copied to the archive form. (`AR_ARCHIVE_NO_ATTACHMENTS`).
`64:`	Diary fields are not copied to the archive form. (`AR_ARCHIVE_NO_DIARY`).

In addition to the archive type, `archiveInfo` also stores the form name (if archive type is `AR_ARCHIVE_FORM`), time to trigger the archive, and the archive qualification criteria. For an archive form, the archive type is `AR_ARCHIVE_NONE` and the name of the source form for which this is the archive form is returned.

auditInfo

If a form is to be audited, this is the audit information for the form. Specify `NULL` for this parameter if you do not want to retrieve the audit information. These are the values for audit type:

`0:`	The selected form is not to be audited. (`AR_AUDIT_NONE`).
`1:`	Audit fields and copy fields on the selected form are copied to the audit shadow form (`AR_AUDIT_COPY`).
`2:`	Audit fields and copy fields on the selected form are copied to the audit log form (`AR_AUDIT_LOG`).

In addition to the audit type, `auditInfo` also stores the form name (if audit type is `AR_AUDIT_COPY` or `AR_AUDIT_LOG`) and the audit qualification criteria.

defaultVui

The label for the default view.

helpText

The help text associated with the form. Specify `NULL` for this parameter if you do not want to retrieve the help text. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

timestamp

A time stamp that specifies the last change to the form (includes changing or adding fields or character menus). Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owning user for the form. Specify `NULL` for this parameter if you do not want to retrieve the owner.

lastChanged

The user who made the last change to the form. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiary

The change diary associated with the form. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to NULL, no properties are returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateSchema, ARDecodeDiary, ARDeleteSchema, ARGetField, ARGetListField, ARGetListAlertUser, ARGetMultipleSchemas, ARSetSchema. See FreeAR for: FreeARCompoundSchema, FreeAREntryListFieldList, FreeARIndexList, FreeARInternalIdList, FreeARPermissionList, FreeARPropList, FreeARSortList, FreeARStatusList.

6.2.106 ARGetServerCharSet

Description

Retrieves a string that represents the name of the character set the API library uses to communicate with the server. If this differs from the client charset, the API converts the data to the right character set.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARGetServerCharSet(
   ARControlStruct*control,
   char*charSet,
   ARStatusList*status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

charSet

A string that names the character set the API library uses to communicate with the server. `charSet` is a buffer whose size is at least `AR_MAX_LANG_SIZE+1`.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetClientCharSet

6.2.107 ARGetServerInfo

Description

Retrieves the requested configuration information for the specified server.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARGetServerInfo(
   ARControlStruct *control,
   ARServerInfoRequestList *requestList,
   ARServerInfoList *serverInfo,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

requestList

A list of one or more server options to return (see the Server options section that follows).

Return values

serverInfo

The information retrieved from the server (see the Server options section that follows). The system returns `NULL` (`AR_DATA_TYPE_NULL`) for server options not retrieved due to error.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

Server options

The server options represent server information about a specific server. The following server options can be retrieved using the `ARGetServerInfo`. If the option is not marked (*get only*), then the server option can be set with the `ARSetServerInfo` function.

- `AR_SERVER_INFO_ACTLINK_DIR`: The name of a directory where all external processes to be run by active links must reside. If the value is `NULL`, active links can run processes located anywhere.
- `AR_SERVER_INFO_ACTLINK_SHELL`: The name of a shell to use when running external processes from an active link. If the value is `NULL`, processes are run from **/bin/sh** (UNIX) or without a shell (Windows).
- `AR_SERVER_INFO_ADMIN_ONLY`: A flag that indicates whether the server is in Administrator Only mode (`T`) or not (`F`). When not in Administrator Only mode, subadministrators can also perform administrator duties. The default value is `F` (Not in Administrator Only mode).
- `AR_SERVER_INFO_ADMIN_OP_PROGRESS` (get only): A flag that indicates whether to report the progress of administrative operations, such as importing. A setting of `0` (the default) means that no progress information is provided, and 1 means that progress is provided. A program can view (but not set) this call. Its value contains progress indicator string if an administrative operation (such as importing) provides one *and* if `AR_SERVER_INFO_ADMIN_OP_TRACKING` was previously set to `1`. Otherwise the value for `AR_SERVER_INFO_ADMIN_OP_PROGRESS` is empty.
  This is a read-only setting. Its value usually remains as an empty string. If `AR_SERVER_INFO_ADMIN_OP_TRACKING` was set to `1` previously; then, during an administrative operation, the server might periodically provide a progress indicator string value here. BMC Remedy AR System API client programs can poll for this setting to track progress of long running operations.
- `AR_SERVER_INFO_ADMIN_OP_TRACKING`: A flag that indicates whether to provide any progress indicative information while an administrative operation (such as import) is taking place. A setting of `0` (the default) means that no progress information is provided, and `1` means that progress information is provided.
- `AR_SERVER_INFO_ALLOW_BACKQUOTE_IN_PROCESS`: A flag that indicates whether backquotes are allowed in a run process command string. A setting of 1 means backquotes are allowed. A setting of `0` means they are disallowed.
- `AR_SERVER_INFO_ALLOW_GUESTS`: A flag that indicates whether the server accepts guest users. Guest users are users not registered with the BMC Remedy AR System. If allowed, guest users have no permissions but are allowed to perform some basic operations. Guest users can submit entries to forms for which permission has been given to the Public group and enter data in fields that have been

defined as allowing any user to submit. If not allowed, unregistered users have no access to the system. Values for this option are 1 (TRUE) and 0 (FALSE). The default value is 1 (allow guest users).

- AR_SERVER_INFO_AP_DEFN_CHECK: The length of time in seconds between checks by an application service to verify that definitions it is using from the BMC Remedy AR System are correct. The value can be from 0 to 3600 seconds (1 hour). A value of **0** means to check definitions with each command. A larger value means a slight delay in recognizing changes to some definition, while a smaller value means the significant overhead of checking the definitions often. The default value is 300 seconds (5 minutes).
- AR_SERVER_INFO_AP_LOG_FILE: The name of the file to use if approval tracing is turned on (see AR_SERVER_INFO_DEBUG_MODE).
- AR_SERVER_INFO_AP_RPC_SOCKET: The RPC program number that the approval server uses when contacting the BMC Remedy AR System. This allows you to define a specific BMC Remedy AR System server for private use by the approval server.
- AR_SERVER_INFO_API_LOG_FILE: The name of the file to use if API tracing is turned on (see AR_SERVER_INFO_DEBUG_MODE).
- AR_SERVER_INFO_APP_SERVICE_PASSWD: The Application Service password. During a get action, this option returns a NULL if there is no password and returns an empty string if there is a password. For security reasons, the actual password string is not returned.
- AR_SERVER_INFO_APPL_PENDING (get only): The name of the form that contains the pending list of application commands to be processed. This form is specific to the use of Application-Commands.
- AR_SERVER_INFO_ARKFORD_LOG_FILE: The name of the file to use if arkford tracing is turned on (see AR_SERVER_INFO_DEBUG_MODE).
- AR_SERVER_INFO_CACHE_DISP_PROP: An integer that indicates whether the server caches just server-only or all display properties by default.

0:	Cache server-only properties (AR_CACHE_DPROP_NONE).
1:	Deprecated, same as AR_CACHE_DPROP_ALL (AR_CACHE_DPROP_VUI).
2:	Deprecated, same as AR_CACHE_DPROP_ALL (AR_CACHE_DPROP_FIELD).
3:	Cache all display properties (AR_CACHE_DPROP_ALL).

- The default value is bits 1 and 2 (cache both VUI and field display properties). The AR_OPROP_CACHE_DISP_PROP property overrides AR_SERVER_INFO_CACHE_DISP_PROP for a specific schema.
- AR_SERVER_INFO_CASE_SENSITIVE (get only): A value that indicates whether the underlying database is case-sensitive. Valid values for this option are 0 (case-sensitive) and 1 (case-insensitive). The default value is 1 (case-insensitive).
  AR_SERVER_INFO_CLUSTERED_INDEX: A value that indicates whether a clustered index is created on the **Entry-Id** field when a new form is created. Valid values for this option are 1 (create a clustered index) and 0 (create a unique index). The default value is 1 (clustered), but you can override this in the BMC Remedy AR System configuration file.
- AR_SERVER_INFO_CONFIG_FILE (get only): The path name of the AR configuration file.
- AR_SERVER_INFO_CURR_PART_DATE_STR (get only): The currency date string.
- AR_SERVER_INFO_CURR_PART_TYPE_STR (get only): The currency type string.

- AR_SERVER_INFO_CURR_PART_VALUE_STR (get only): The currency value string.
- AR_SERVER_INFO_CURRENCY_INTERVAL: The time interval, in minutes, before client programs refreshes currency ratios. A value of zero means never refresh.
- AR_SERVER_INFO_DB_CHAR_SET (get only): The character set of the BMC Remedy AR System database.
- AR_SERVER_INFO_DB_CONNECTION_RETRIES: An integer that indicates the number of times the BMC Remedy AR System server tries to connect to the database.
- AR_SERVER_INFO_DB_MAX_ATTACH_SIZE: The maximum size (in bytes) allowed for attached files in an Oracle database. The maximum valid value for this option depends on the database server operating system and configuration. The default value is 2 GB.
- AR_SERVER_INFO_DB_MAX_TEXT_SIZE: The maximum size (in bytes) allowed for long character text data in an Oracle, Microsoft SQL Server, or Sybase database. For an Oracle database, this value is also used for memory allocation during the processing of long text data, so if it is set too high, the database storage used by long text data might be very large. The default value for Oracle is 1 MB. For SQL Server and Sybase, the default and maximum values are 2,147,483,647 bytes.
- AR_SERVER_INFO_DB_NAME (get only): The name of the underlying SQL database (*not* applicable to Oracle databases). The default value is ARSystem.
- AR_SERVER_INFO_DB_TYPE (get only): The underlying database type (character string).
- AR_SERVER_INFO_DB_USER (get only): The user name BMC Remedy AR System uses to access the underlying database.
- AR_SERVER_INFO_DB_VERSION (get only): The underlying database version (character string).
- AR_SERVER_INFO_DB2_DB_ALIAS (get only): IBM DB2 database alias name.
- AR_SERVER_INFO_DB2_SERVER (get only): Name of the server where the DB2 database resides.
- AR_SERVER_INFO_DBCONF (get only): The complete contents of the **db.conf** (**ardb.cfg**) file.
- AR_SERVER_INFO_DBHOME_DIR (get only): The home directory for the underlying SQL database.
- AR_SERVER_INFO_DEBUG_MODE: A bitmask that indicates the server debug modes.

Bit 0:	Enables SQL tracing. The default file for SQL tracing is arsql.log (see AR_SERVER_INFO_SQL_LOG_FILE).
Bit 1:	Enables filter tracing. The default file for filter tracing is arfilter.log (see AR_SERVER_INFO_FILTER_LOG_FILE).
Bit 2:	Enables user tracing. The default file for user tracing is aruser.log (see AR_SERVER_INFO_USER_LOG_FILE).
Bit 3:	Enables escalation tracing. The default file for escalation tracing is arescl.log (see AR_SERVER_INFO_ESCALATION_LOG_FILE).
Bit 4:	Enables API tracing. The default file for API tracing is arapi.log (see AR_SERVER_INFO_API_LOG_FILE).
Bit 5:	Enables thread tracing. The default file for thread tracing is arthread.log (see AR_SERVER_INFO_THREAD_LOG_FILE).
Bit 6:	Enables alert tracing for the arserverd process. The default file for alert tracing is aralert.log (see AR_SERVER_INFO_ALERT_LOG_FILE).

Bit 7:	Enables arforkd tracing for the arserverd process. The default file for arforkd tracing is arforkd.log (see `AR_SERVER_INFO_ARFORK_LOG_FILE`).
Bit 8:	Enables server group tracing. The default file for server group tracing is arsrvgrp.log (see `AR_SERVER_INFO_SERVERGROUP_LOG_FILE`).
Bit 15:	Enables distributed server tracing (only applicable to Distributed Server Option (DSO)). The default file for distributed server tracing is ardist.log (see `AR_SERVER_INFO_DS_LOG_FILE`).
Bit 16:	Enables approval server debugging.
Bit 17:	Enables plug-in tracing (only applicable to external data sources). The default file for plug-in tracing is **arplugin.log** (see `AR_SERVER_INFO_PLUGIN_LOG_FILE`).

- `AR_SERVER_INFO_DEFAULT_ORDER_BY`: An integer (`1` or `0`) that indicates the default order of requests in a form when issuing `ARGetListEntry` calls, so that the sorting is done on the database and before the results are returned. Valid values for this option are `0` (no sort order) or `1` (use EntryId order). The default value is `1`.
- `AR_SERVER_INFO_DELAYED_CACHE`: A flag that indicates whether to disable recaching of the shared cache after a time-out (`T`) or not (`F`). The default value is `F` (delayed recache not disabled).
- `AR_SERVER_INFO_DFLT_ALLOW_CURRENCIES`: Default list of allowable currencies when new currency fields are created.
- `AR_SERVER_INFO_DFLT_FUNC_CURRENCIES`: Default list of functional currencies when new currency fields are created.
- `AR_SERVER_INFO_DISABLE_ADMIN_OPERATIONS`: The value that indicates whether certain operations performed only by administrators and subadministrators are disabled. Valid values for this option are `1` (Disabled) and `0` (Enabled).
- `AR_SERVER_INFO_DISABLE_ARCHIVE`: The value that indicates whether archiving is disabled. Valid values for this option are `1` (Disabled) and `0` (Enabled).
- `AR_SERVER_INFO_DISABLE_FTS_INDEXER`: The value that indicates whether the full text search (FTS) indexer is disabled. Valid values for this option are `1` (Disabled) and `0` (Enabled).
- `AR_SERVER_INFO_DISABLE_NON_UNICODE_CLIENTS`: A flag that indicates whether a Unicode server declines calls from non-Unicode clients. Regardless of the setting, however, calls from BMC Remedy Alert* and BMC Remedy Developer Studio are always accepted. Valid values are as follows:

`0:`	The Unicode server responds to calls from all clients.
`nonzero:`	The Unicode server returns error status AR_RETURN_ERROR and corresponding error message AR_ADMIN_ONLY_MODE to calls from non-Unicode clients.

  * BMC Remedy Alert is no longer shipped with BMC Remedy AR System.
- `AR_SERVER_INFO_DS_LOG_FILE`: The name of the file to use if distributed server tracing is turned on (see `AR_SERVER_INFO_DEBUG_MODE`).Valid values for this option are `1` (`TRUE`) and `0` (`FALSE`).
- `AR_SERVER_INFO_DS_MAPPING` (get only): The name of the form that contains distributed mapping definitions (only applicable to DSO only).
- `AR_SERVER_INFO_DS_PENDING` (get only): The name of the form that contains the pending operation list (only applicable to DSO).

- AR_SERVER_INFO_DS_RPC_SOCKET: The specific server socket to use for the distributed server. Valid values for this option are 390600 and 390680 through 390694. Any other value causes the distributed server to use the default server.
- AR_SERVER_INFO_DS_SVR_LICENSE: The distributed server license type (character string).
- AR_SERVER_INFO_DSO_DEST_PORT: The TCP port that the distributed server option uses to communicate with the BMC Remedy AR System server.
- AR_SERVER_INFO_DSO_ERROR_RETRY: An integer that specifies the error retry behavior for DSO.

0:	Standard connection and transmission errors qualify for retry. This is the default setting (see AR_DSO_ERROR_RETRY_STANDARD) .
1:	No errors qualify for retry (see AR_DSO_ERROR_RETRY_NO_RETRY).
2:	All errors qualify for retry (see AR_DSO_ERROR_RETRY_ALL_RETRY).
3:	Standard connection and transmission errors plus database errors qualify for retry (see AR_DSO_ERROR_RETRY_STANDARD_DB).

- AR_SERVER_INFO_DSO_MAX_QUERY_SIZE: The maximum number of entries that the Distributed Server Option (DSO) retrieves in a single query to the server. The minimum value for this option is 1, and there is no upper boundary.
- AR_SERVER_INFO_DSO_MERGE_STYLE: An integer (1 or 0) that indicates the merge style to use for the DSO. A 0 indicates the new style; 1 indicates the old style. The default is 0. For a 1 value (old style), the merge style overwrites entries; for a 0 value (new style), a true merge is performed according to the rules of the DSO setup. For backward compatibility, this server option can be set to 1 .
- AR_SERVER_INFO_DSO_RPCPROG_NUM: The RPC program number settings that DSO uses to access the local server. The ranges available for private queues are:
  - 390600
  - 390621 to 390634
  - 390636 to 390669
  - 390680 to 390694
- AR_SERVER_INFO_DSO_TARGET_PASSWD: The password for the target DSO server. During a get action, this option returns a list of servers that have been assigned a password. For security reasons, the actual password strings are not returned.
- AR_SERVER_INFO_DSO_TIMEOUT_NORMAL: The number of seconds the DSO server waits before disconnecting inactive clients. The range of values is 60 to 21600 (60 seconds to 6 hours). The default value is 120.
- AR_SERVER_INFO_DSO_USER_PASSWD: The password for the DSO server. During a get action, this option returns a NULL if there is no password and returns an empty string if there is a password. For security reasons, the actual password string is not returned.
- AR_SERVER_INFO_EA_GROUP_MAPPING: Specifies mapping of LDAP groups to BMC Remedy AR System groups for the purpose of external authentication. The value of this option consists of one or more pairs of group names separated by semicolons, as shown in the following example:
  "LDAP-1" "ARS-1";"LDAP-2" "ARS-2";"LDAP-3" "ARS-3"
  Use mapping as an alternative to creating an BMC Remedy AR System group corresponding to each

LDAP group. You can map each LDAP group to an BMC Remedy AR System group (as in the example) or multiple LDAP groups to a single BMC Remedy AR System group. (If you try to map a single LDAP group to multiple BMC Remedy AR System groups, only the first mapping expression is valid.)
This option can be combined with `AR_SERVER_INFO_EA_IGNORE_EXCESS_GROUPS`, if desired.

- `AR_SERVER_INFO_EA_IGNORE_EXCESS_GROUPS`: A flag used by BMC Remedy AR System during external authentication. A value of `0` indicates that the user is authenticated when there is a match between every LDAP group to which the user belongs and a corresponding group in BMC Remedy AR System. A value of `1` indicates that the user is authenticated when any single LDAP group to which the user belongs matches a group in BMC Remedy AR System, and the excess LDAP groups are ignored. The default value is `0`.
  This option can be combined with `AR_SERVER_INFO_EA_GROUP_MAPPING`, if desired.
- `AR_SERVER_INFO_EA_RPC_SOCKET`: The RPC socket number on which an external authentication server awaits requests for authentication. A `0` value means external authentication is not used. This attribute persists in the **ar.cfg /ar.conf** file. The default value is `0`.
- `AR_SERVER_INFO_EA_RPC_TIMEOUT`: The RPC timeout (seconds) used when making calls to the authentication (AREA) server (for example, 30 seconds). This attribute persists in the **ar.cfg/ar.conf** file. The default value is `30` seconds.
- `AR_SERVER_INFO_EA_SYNC_TIMEOUT`: The internal (seconds) the BMC Remedy AR System server uses to periodically call the `AREANeedToSyncCallback()` function on the external authentication server, which instructs the BMC Remedy AR System server to renew its internally stored user information in the event there are changes made to the source used to authenticate users. A `0` value means that the BMC Remedy AR System server does not call the external authentication (AREA) server. This attribute persists in the **ar.cfg /ar.conf** file. The default value is `300` seconds.
- `AR_SERVER_INFO_EMAIL_AIX_USE_OLD_EMAIL`: A flag that indicates whether to use the old email or the new Email engine. A zero (`0`) setting indicates that the new email is used. A setting of one (`1`) indicates that the old email is used. (AIX only)
- `AR_SERVER_INFO_EMAIL_FROM`: The sender name to use for filter-generated email notifications where no subject is specified. Only trusted email users can use this name (see documentation about the **/etc/sendmail.cf** file for more information about trusted users). (UNIX only)
- `AR_SERVER_INFO_EMAIL_IMPORT_FORM`: A flag that indicates whether to enable or disable the import of the email forms. If set to zero (0), the forms are not imported. If set to one (1), the forms are imported.
- `AR_SERVER_INFO_EMAIL_LINE_LEN`: The maximum line length of e-mail messages.
- `AR_SERVER_INFO_EMAIL_SYSTEM`: A character string that indicates the email system on UNIX.
- `AR_SERVER_INFO_EMAIL_TIMEOUT`: (UNIX only) The length of time in seconds before the system closes the pipe to `sendmail` to unblock the BMC Remedy AR System server. Valid values for this option are from `1` to `300`. The default value is `10`.
- `AR_SERVER_INFO_EMBEDDED_SQL` (get only): A value that indicates whether the underlying SQL database is embedded (bundled with BMC Remedy AR System). Valid values for this option are `0` (not embedded) and `1` (embedded).
- `AR_SERVER_INFO_ERROR_EXCEPTION_LIST`: A list of error codes that cause diagnostic output to be written to the error log when the server encounters the errors.

- AR_SERVER_INFO_ESCALATION_LOG_FILE: The name of the file to use if escalation tracing is turned on (see AR_SERVER_INFO_DEBUG_MODE).
- AR_SERVER_INFO_EXCEPTION_OPTION: A value that indicates the diagnostic output option for exceptions. A setting of 0 indicates that a stack trace is to be included. A setting of 1 indicates that a stack trace is not to be included.
- AR_SERVER_INFO_EXPORT_VERSION (get only): An integer that indicates the server's export version number.

3:	Export version of BMC Remedy AR System 3.x
4:	Export version of BMC Remedy AR System 4.0.3
5:	Export version of BMC Remedy AR System 4.5
6:	Export version of BMC Remedy AR System 5.0
7:	Export version of BMC Remedy AR System 5.1
8:	Export version of BMC Remedy AR System 6.0 and BMC Remedy AR System 6.3
9:	Export version of BMC Remedy AR System 7.0
10:	Export version of BMC Remedy AR System 7.1
11:	Export version of BMC Remedy AR System 7.5

- AR_SERVER_INFO_EXT_AUTH_CAPABILITIES: A value that indicates the return data capabilities for the AREA plug-in in use. This setting does not control the AREA plug-in. It only describes the behavior of the plug-in, allowing for server optimization.

1:	No email address (see AR_EXT_AUTH_DATA_NO_EMAIL_ADDR).
2:	No notify mechanism (see AR_EXT_AUTH_DATA_NO_NOTIFY_MECH).
4:	No group identifiers (see AR_EXT_AUTH_DATA_NO_GROUP_IDS).
8:	No license information (see AR_EXT_AUTH_DATA_NO_LICENSE_INFO).

- AR_SERVER_INFO_FILT_MAX_STACK: The maximum number of levels of recursion allowed for a given operation. The data modification performed by an AR_FILTER_ACTION_FIELDP filter action could trigger a second set, or level, of filters, one of which could trigger filters a third level down and so on. This option limits the number of times such recursion can happen, preventing the server crash that would occur if the recursion continued indefinitely. The default value is 25.
- AR_SERVER_INFO_FILT_MAX_TOTAL: The maximum number of filters that the server executes for a given operation. The default value is 10000.
- AR_SERVER_INFO_FILTER_API_RPC_TIMEOUT: The time limit, in seconds, for the filter API RPC to respond to the server's request before an error is returned. The default value is 40 seconds. The minimum value is 1, and the maximum value is 300.
- AR_SERVER_INFO_FILTER_LOG_FILE: The name of the file to use if filter tracing is turned on (see AR_SERVER_INFO_DEBUG_MODE).
- AR_SERVER_INFO_FIXED_LICENSE (get only): The number of fixed user licenses (int).

- AR_SERVER_INFO_FLASH_DAEMON: A flag that indicates whether Flashboards is installed (T) or not (F). The default value is 1 if the server has one or more Flashboards licenses, and 0 if the server has no licenses.
- AR_SERVER_INFO_FLOAT_LICENSE (get only): The number of floating write licenses defined.
- AR_SERVER_INFO_FLOAT_TIMEOUT: The number of hours the server waits before disconnecting inactive users. If a user is holding a floating write license token, the system also frees the token at this time. The default value is 2 hours.
- AR_SERVER_INFO_FLUSH_LOG_LINES: A value (1 (TRUE) and 0 (FALSE)) that indicates whether the server flushes every line written to a log to disk. If the line is not flushed to disk, the line is buffered by the file system.
- AR_SERVER_INFO_FT_CASE_SENSITIVITY: A flag that specifies case sensitivity for the server's full text option. A value of 0 indicates that full text searching is case-sensitive. A value of 1 indicates that full text searching is not case-sensitive. The default value is 1.
- AR_SERVER_INFO_FT_COLLECTION_DIR: The location in the file system where full text indexes are stored.
- AR_SERVER_INFO_FT_CONFIGURATION_DIR: The location in the file system where search engine configuration files are stored.
- AR_SERVER_INFO_FT_DISABLE_SEARCH: A value that indicates the server's full text searching state. Valid settings are 0 (full text searching enabled) and 1 (full text searching disabled). The default value is 0.
  Disabling full text searching applies only to field-based searching. It does not apply to multi-form searching.
- AR_SERVER_INFO_FT_FORM_REINDEX: A value that specifies reindexing all full text indexed fields on specified forms. When using this option for indexing, supply a character string containing the form names separated by a form-feed character ('\f'). On retrieval, this option returns the list of forms separated by a form-feed character where indexing is still pending. (The AR_SERVER_INFO_FT_REINDEX option specifies reindexing of *all* of the forms on the server. When re-indexing tasks are pending, there is no distinction between how they originated when returning the operation status.)
- AR_SERVER_INFO_FT_RECOVERY_INTERVAL: The number of minutes the server waits between periodic attempts to index entries that unexpectedly failed to index in a prior attempt. The default value is 60.
- AR_SERVER_INFO_FT_REINDEX: A value that specifies reindexing *all* of the forms that have full text indexed fields on the server. A setting of 1 initiates the reindex operation. On retrieval, this option is set to 1 if a reindex operation is in progress. Otherwise, the setting is 0. (The AR_SERVER_INFO_FT_FORM_REINDEX option specifies reindexing of all full text indexed fields on specified forms. When re-indexing tasks are pending, there is no distinction between how they originated when returning the operation status.)
- AR_SERVER_INFO_FT_SEARCH_MATCH_OP: The server's match option for full text search match option. Valid settings are as follows:

0:	Force leading and trailing wildcard.
1:	Force trailing wildcard.
2:	Ignore leading wildcard.

3:	Remove wildcards.
4:	Unchanged.

The default value is 4.

- AR_SERVER_INFO_FT_SIGNAL_DELAY: A value that specifies the number of seconds before a signal that needs an index is sent to the server that owns the full text indexing operation in a server group (if no signal is pending). The default value is 10 seconds.
- AR_SERVER_INFO_FT_STOP_WORDS: A list of words to be ignored during indexing. The words are separated by semicolons.
- AR_SERVER_INFO_FT_TEMP_DIR: The location where temporary files associated with the search engine are stored.
- AR_SERVER_INFO_FT_THRESHOLD_HIGH: During the processing of full-text-search data, the server combines results from sub-queries to generate a final set of results. If the number of rows created during processing exceeds this value, the server returns an error that indicates that the search is too complex. The default value is 1,000,000.
- AR_SERVER_INFO_FT_THRESHOLD_LOW: During the processing of full-text-search data, the server creates a temporary table if the number of matches equals or exceeds this value. If the number of matches is under this value, the server uses the SQL IN operator for a query on an existing table. The default value is 200.
- AR_SERVER_INFO_FTEXT_FIXED (get only): The number of fixed FTS licenses defined.
- AR_SERVER_INFO_FTEXT_FLOAT (get only): The number of floating FTS licenses defined.
- AR_SERVER_INFO_FTEXT_TIMEOUT: The number of hours the server waits before disconnecting inactive users with FTS licenses. If a user is holding a floating FTS license token, the system also frees the token at this time.
- AR_SERVER_INFO_FTINDEXER_LOG_FILE: The file name used for the full text indexer log.
- AR_SERVER_INFO_FULL_HOSTNAME (get only): The full DNS host name of the server ("long" machine name).
- AR_SERVER_INFO_G_CACHE_CHANGE (get only): The time of the last change to the group cache.
- AR_SERVER_INFO_GUESTS_RESTRICT_READ: Specifies whether guest users are granted a restricted read license. Values for this option are 1 (TRUE) or 0 (FALSE). The default value is 0.
- AR_SERVER_INFO_GUID_PREFIX: The character string used as a prefix for generated GUIDs.
- AR_SERVER_INFO_HARDWARE (get only): The server hardware type (character string).
- AR_SERVER_INFO_HOMEPAGE_FORM: The server homepage form (character string).
- AR_SERVER_INFO_HOSTNAME (get only): The host name of the server ("short" machine name).
- AR_SERVER_INFO_INFORMIX_DBN* (get only): Informix database server name. This option has a value only if you are using an Informix database.
- AR_SERVER_INFO_INFORMIX_RELAY_MOD* (get only): A character string that indicates the Informix relay module.
- AR_SERVER_INFO_INFORMIX_TBC* (get only): The Informix configuration file. This option has a value only if you are using an Informix database. The default value is onconfig.

(Options marked with a * are available for Informix database, which is no longer supported in BMC Remedy AR System.)

- AR_SERVER_INFO_IP_NAME: A string that contains multiple names for the server, separated by semicolons.
- AR_SERVER_INFO_JAVA_VM_OPTIONS: The Java Virtual Machine (JVM) options.
- AR_SERVER_INFO_LOCALIZED_SERVER: A variable that indicates whether the server is localized. Values for this option are 1 (TRUE) and 0 (FALSE). The default value is 1 (server is localized).
- AR_SERVER_INFO_LOGFILE_APPEND: A value (1 (TRUE) or 0 (FALSE)) that indicates whether to create a separate ***.bak** file or to append to the existing log file. A 0 value creates a ***.bak** file; a 1 value indicates that new log information be appended to the existing file. The default value is 0 (creates a ***.bak** file).
- AR_SERVER_INFO_MAX_ENTRIES: The maximum number of entries returned by a single query. Because users can also specify the maximum number of entries returned (in query preferences), the actual maximum is the lower of these two values. The default value is no (server-defined) maximum.
- AR_SERVER_INFO_MAX_LOG_FILE_SIZE: The maximum size for system log files. The default value is 0 (no limit).
- AR_SERVER_INFO_MAX_PASSWORD_ATTEMPTS: The maximum number of times that someone can attempt to log in with an invalid password. If this number is exceeded, the account is locked until the password is reset by the user or an administrator. Valid values are 0 (unlimited) or any positive integer. The default value is 0.
- AR_SERVER_INFO_MAX_SCHEMAS (get only): The maximum number of forms allowed. The default value is 0 (no limit).
- AR_SERVER_INFO_MESSAGE_CAT_SCHEMA (get only): The name of the BMC Remedy Message Catalog form.
- AR_SERVER_INFO_MID_TIER_PASSWD: The password that the mid tier uses to access the BMC Remedy AR System server. During a get action, this option returns a NULL if there is no password and returns an empty string if there is a password. For security reasons, the actual password string is not returned.
- AR_SERVER_INFO_MINIMUM_API_VER: A value that indicates the minimum API version that the server supports. The API versions are:

5:	API version of BMC Remedy AR System 3.x
6:	API version of BMC Remedy AR System 4.0.3
7:	API version of BMC Remedy AR System 4.5
8:	API version of BMC Remedy AR System 5.0
9:	API version of BMC Remedy AR System 5.1
10:	API version of BMC Remedy AR System 6.0
11:	API version of BMC Remedy AR System 6.3
12:	API version of BMC Remedy AR System 7.0
13:	API version of BMC Remedy AR System 7.1
14:	API version of BMC Remedy AR System 7.5

- `AR_SERVER_INFO_MINIMUM_CMDB_API_VER`: A value that indicates the minimum CMDB API version that the server supports. The API versions are:

`3:`	API version of BMC Atrium CMDB 2.0 and 2.0.01
`4:`	API version of BMC Atrium CMDB 2.1

- `AR_SERVER_INFO_MULTI_SERVER` (get only): An integer (`0` /`1`) value that indicates the use of the multi-server option. A `0` value indicates a single server; a `1` value indicates the use of the multi-server option.
- `AR_SERVER_INFO_MULTIPLE_ARSYSTEM_SERVERS`: A flag that indicates that more than one server is running on the host system. Valid values are `1` (`TRUE`) and `0` (`FALSE`). The default value is `FALSE`. For more information about configuring multiple servers on one machine, see Installing multiple instances of BMC Remedy AR System on one computer.
- `AR_SERVER_INFO_NEXT_ID_BLOCK_SIZE`: Specifies the block size at which `entryID` s for all schemas are allocated. This setting is used to enhance the performance of the `CREATE ENTRY` operation.
  Values for this option are as follows:

`1:`	EntryID values increase consistently by one, and there are no gaps in the entryID sequence.
`2 to 1000:`	EntryID s are allocated in blocks of this size.

  The default value is `1`. If an invalid setting is encountered, the option is forced to a setting of `1`.
  You can override `AR_SERVER_INFO_NEXT_ID_BLOCK_SIZE` for a specific schema by setting the `Next Request ID Block Size` form property in BMC Remedy Developer Studio.
  `AR_SERVER_INFO_NEXT_ID_BLOCK_SIZE` does not work on the "distributed pending" schema. Correct operation of Distributed Server Option (DSO) requires **entryID**s to be specified in the order in which they were written.
  Use of `AR_SERVER_INFO_NEXT_ID_BLOCK_SIZE` might result in gaps in the **entryID** sequence. A gap occurs if an **entryID** transaction completes successfully, but the remainder of the operation is rolled back. The **entryID** would be lost because it cannot be returned to the system.
- `AR_SERVER_INFO_NEXT_ID_COMMIT`: During a `CREATE ENTRY` operation for a schema, specifies whether to commit the current transaction immediately after an **entryID** is retrieved and updated, and start a new one for the remainder of the operation. Values are `1` (`TRUE`) and `0` (`FALSE`). If this setting is not specified in the **ar.conf** file, the default value is `FALSE`.
  If this option is set to `TRUE`, then during creation of an entry, the existing transaction is committed and a new transaction is started immediately after the **entryID** has been assigned. This behavior allows other pending transactions to proceed sooner.
  `AR_SERVER_INFO_NEXT_ID_COMMIT` provides a small performance improvement at the expense of potential gaps in the **entryID** sequence. A gap occurs if an **entryID** transaction completes successfully, but the remainder of the operation is rolled back. The **entryID** would be lost because it cannot be returned to the system.

Instead of AR_SERVER_INFO_NEXT_ID_COMMIT, you might prefer to use AR_SERVER_INFO_NEXT_ID_BLOCK_SIZE. The latter option is more flexible and offers a more significant enhancement in performance.

- AR_SERVER_INFO_NFY_TCP_PORT: The TCP port that the server uses.
- AR_SERVER_INFO_NOTIFY_WEB_PATH: The default WWW path for URL shortcuts embedded in e-mail notifications.
- AR_SERVER_INFO_ORACLE_BULK_FETCH_COUNT: Defines the number of the rows of data fetched at a time from the result set when querying an Oracle database. The minimum is 1, the maximum is 100, and the default is 50. Higher values cause the BMC Remedy AR System server to use more memory but make fewer database fetches during data retrieval.
- AR_SERVER_INFO_ORACLE_CLOB_STORE_INROW: Specifies whether CLOBs in the Oracle database are stored IN ROW. Values for this option are 0 (DISABLE STORAGE IN ROW) and 1 (ENABLE STORAGE IN ROW). The default value is 0.
  It is recommended that diary fields and character fields with unlimited length be created with IN ROW storage disabled.
- AR_SERVER_INFO_ORACLE_SID (get only): The system ID of the Oracle database you are accessing. This option has a value only if you are using an Oracle database.
- AR_SERVER_INFO_ORACLE_TWO_T (get only): The two-task environment setting for remote access to an Oracle database. This option has a value only if you are using an Oracle database.
- AR_SERVER_INFO_OS (get only): A character string that indicates the server operating system that includes the version number.
- AR_SERVER_INFO_PER_THREAD_LOGS: A value (1 (TRUE) and 0 (FALSE)) that indicates whether the server creates a log for each worker thread. A value of zero means the worker threads use the shared logs. A value of 1 causes each thread to create its own copy of the shared logs.
- AR_SERVER_INFO_PLUGIN_ALIAS: A list of plug-in aliases.
- AR_SERVER_INFO_PLUGIN_DEFAULT_TIMEOUT: The number of seconds the server waits for a response from the plug-in server. The range of values is 60 to 600 (60 seconds to 10 minutes). The default value is 60.
- AR_SERVER_INFO_PLUGIN_LOG_FILE: The name of the plug-in service log file (see AR_SERVER_INFO_DEBUG_MODE).
  For information about the plug-in logging feature, see Running the plug-in server.
- AR_SERVER_INFO_PLUGIN_LOG_LEVEL: An integer that specifies the logging level for plug-ins. Values range from 100 to 10000. The default value is 10000 (OFF).
  For descriptions of the predefined logging levels, see Running the plug-in server.
- AR_SERVER_INFO_PLUGIN_PASSWD: The plug-in service password. During a get action, this option returns a NULL if there is no password and returns an empty string if there is a password. For security reasons, the actual password string is not returned.
- AR_SERVER_INFO_PLUGIN_PORT: An integer that specifies the port of the plug-in server. A value of 0 for the port means no port was specified.
- AR_SERVER_INFO_PLUGIN_TARGET_PASSWD: The password that the BMC Remedy AR System server uses to communicate with a plug-in service that runs at the host name and port number specified. During a get action, this option returns a list of servers that have been assigned a password. For security reasons, the actual password strings are not returned.

- AR_SERVER_INFO_PREF_SERVER_OPTION: An integer that specifies the preference server behavior for this server, to be honored by the client. The default value is 1.

1:	User-defined value.
2:	Always use this server.
3:	Do not use this server.

- AR_SERVER_INFO_PS_RPC_SOCKET: The RPC program number and port pairs the private queues use. A value of 0 for the port means no port was specified.
- AR_SERVER_INFO_REGISTER_PORTMAPPER: A flag that indicates whether to register with BMC Remedy AR System Portmapper (T) or not (F). The default value is 1.
- AR_SERVER_INFO_REM_SERV_ID (get only): The BMC Remedy AR System server ID associated with the license (character string).
- AR_SERVER_INFO_RPC_CLIENT_XDR_LIMIT: An integer that specifies the maximum size in bytes of RPC data a client can send to the BMC Remedy AR System server. The default value is 0, which means there is no limit.
- AR_SERVER_INFO_RPC_NON_BLOCKING_IO: A flag that indicates whether the server is in RPC non blocking I/O mode or not. Valid values are 1 (TRUE) and 0 (FALSE). The default value is 0 (FALSE).
- AR_SERVER_INFO_SAVE_LOGIN: A value that indicates whether users must log in to clients.

0:	Controlled by user (default setting).
1:	Force no login (controlled by the administrator).
2:	Force login (controlled by the administrator).

- AR_SERVER_INFO_SCC_COMMENT_CHECKIN: An integer (0 or 1) value that indicates whether a source code control integration requires you to enter a comment at check-in time. The default 1 value means no comment is required.
- AR_SERVER_INFO_SCC_COMMENT_CHECKOUT: An integer (0 or 1) value that indicates whether a source code control integration requires you to entire a comment at checkout time. A default 1 value means no comment is required.
- AR_SERVER_INFO_SCC_ENABLED: A value that indicates whether a source code control system is being used with the BMC Remedy AR System. An F value (default) means a source code control system is not present.
- AR_SERVER_INFO_SCC_INTEGRATION_MODE: An integer (T or F) value that indicates the level of source code control integration. An F (the default) value indicates an advisory level of integration, which means that you can modify and save an object without having it checked out from the source code control application. A T value means enforced integration, which has strict rules for controlling source files (for example, enforced integration means that other administrators are not able to modify and save changes on an object that you have checked out).
- AR_SERVER_INFO_SCC_PROVIDER_NAME: A character string for the source code control system provider name. If none is present, this value is NULL. This string is limited to 255 characters.
- AR_SERVER_INFO_SCC_TARGET_DIR: A character string for the source code control system target directory. If none is present, this value is NULL. This string is limited to 255 characters.

- `AR_SERVER_INFO_SERVER_DIR` (get only): The data directory for the BMC Remedy AR System. This directory contains support files for the underlying SQL database (but no actual database files). For flat file databases, this directory contains both database definition and data files.
- {{AR_SERVER_INFO_SERVER_IDENT }}(get only): The unique identifier for the server (character string).
- `AR_SERVER_INFO_SERVER_LANG`(get only): The local language setting of the server.
- `AR_SERVER_INFO_SERVER_LICENSE`(get only): The server license type (character string).
- `AR_SERVER_INFO_SERVER_NAME`: An alias for the current server name. The BMC Remedy AR System server provides this name when it receives a request for the server name. The value for this field is the name without the domain extension. The BMC Remedy AR System server automatically appends the current domain name to the name. For example, if the server is in the domain **arrow.com**, and the value for this setting is `alpha`, the name is **alpha.arrow.com**. Note that a setting of **alpha.arrow.com** results in an unrecognizable name: **alpha.arrow.com.arrow.com**.
  See also `Server-Name` in the **ar.conf** file.
- `AR_SERVER_INFO_SERVERGROUP_INTERVAL`: The interval (in seconds) for server group checking.
- `AR_SERVER_INFO_SERVERGROUP_LOG_FILE`: The file name used for the server group log.
- `AR_SERVER_INFO_SERVERGROUP_MEMBER`: The value that indicates whether the server is a server group member.

`0:`	Not a server group member.
`1:`	A server group member.

- `AR_SERVER_INFO_SERVERGROUP_NAME`: The server group name.
- `AR_SERVER_INFO_SET_PROC_TIME`: The number of seconds the server waits before ending a Set Fields process that has not completed. Valid values for this option are `1` through `20`. The default value is five seconds.
- `AR_SERVER_INFO_SG_AIE_STATE` (get only): The server group Atrium Integration Engine state.

`0:`	No server group, or operation is not managed.
`1:`	Suspended.
`2:`	Resumed.

- AR_SERVER_INFO_SG_EMAIL_STATE (get only): The server group email state.

`0:`	No server group, or operation is not managed.
`1:`	Suspended.
`2:`	Resumed.

- `AR_SERVER_INFO_SG_FLASHBOARDS_STATE` (get only): The server group flashboards state.

`0:`	No server group, or operation is not managed.
`1:`	Suspended.
`2:`	Resumed.

- AR_SERVER_INFO_SQL_LOG_FILE: The name of the file to use if SQL tracing is turned on (see AR_SERVER_INFO_DEBUG_MODE).
- AR_SERVER_INFO_SSTABLE_CHUNK_SIZE: The maximum number of entries fetched during server side table processing. Zero (0) is unlimited.
- AR_SERVER_INFO_STRUCT_CHANGE (get only): The last structure change (for this run of the server).
- AR_SERVER_INFO_SUBMITTER_MODE: A value that indicates whether the Submitter field can be changed and whether the submitter of an entry must have a license to modify it. Valid values for this option are 1 (locked) and 2 (changeable). The default value is 2 (changeable).
  In *locked* mode:
  - the **Submitter** field cannot be changed after submit.
  - the submitter with or a read or a write license, but not with a restricted read license, can modify the entry (if the Submitter group has Change permission).
    In *changeable* mode, the **Submitter** field can be changed after submit, but the submitter must have a write license to modify the entry (if the Submitter group has Change permission).
- AR_SERVER_INFO_SUPPRESS_WARN: A series of zero or more message numbers (separated by spaces) that identify the informational or warning messages that the system suppresses.
- AR_SERVER_INFO_SVR_EVENT_LIST: A character string that consists of a list of numeric event types separated by semicolons. If this tag is not set or does not include event types, the server does not create an event schema or record schema events. If the server does not recognize an event type, it ignores that element. When this option has a value of NULL, the server removes the event list.

1:	AR_SVR_EVENT_CHG_SCHEMA	Logs changes to schemas.
2:	AR_SVR_EVENT_CHG_FIELD	Logs changes to fields.
3:	AR_SVR_EVENT_CHG_CHARMENU	Logs changes to character menus.
4:	AR_SVR_EVENT_CHG_FILTER	Logs changes to filters.
5:	AR_SVR_EVENT_CHG_IMPORT	Logs changes to imports.
6:	AR_SVR_EVENT_CHG_ACTLINK	Logs changes to active links.
7:	AR_SVR_EVENT_CHG_ESCAL	Logs changes to escalations.
8:	AR_SVR_EVENT_CHG_VUI	Logs changes to VUIs.
9:	AR_SVR_EVENT_CHG_CONTAINER	Logs changes to containers.
10:	AR_SVR_EVENT_CHG_USERS	Logs the users added, modified, or deleted.
11:	AR_SVR_EVENT_CHG_GROUPS	Logs the groups added, modified, or deleted.
12:	AR_SVR_EVENT_CHG_SVR_SETTINGS	Logs changes to server settings.
13:	AR_SVR_EVENT_CHG_ALERT_USERS	Logs changes to user alert messages.
14:	AR_SVR_EVENT_ARCHIVE_DONE	Logs the status of archive for a form.
15:	AR_SVR_EVENT_SERVGROUP_ACTION	Logs server group events.

- `AR_SERVER_INFO_SVR_STATS_REC_INTERVAL`: The time interval, in seconds, that the server records statistics into the server statistics form.
- `AR_SERVER_INFO_SVR_STATS_REC_MODE`: An integer that specifies the type of recording modes that the server uses to record server statistics. This determines if statistics are recorded in the server statistics form.

`0:`	`AR_SVR_STATS_RECMODE_OFF`	The server does not record statistics.
`1:`	`AR_SVR_STATS_RECMODE_CUMUL_ONLY`	The server records only cumulative queue statistics.
`2:`	`AR_SVR_STATS_RECMODE_CUMUL_QUEUE`	The server records both cumulative and individual queue statistics.

- `AR_SERVER_INFO_SYBASE_CHARSET` (get only): The character set being used to access a Sybase database. This option has a value only if you are using a Sybase database.
- `AR_SERVER_INFO_SYBASE_SERV` (get only): The Sybase logical server name. This option has a value only if you are using a Sybase database. The default value is `SYBASE`.
- `AR_SERVER_INFO_SYS_LOGGING_OPTIONS`: A value that indicates the logging options for operating system logging.

`1:`	Suppress logging to the system log file (see `AR_SYSTEM_LOGGING_OPTION_NONE`).

- `AR_SERVER_INFO_TCD_TCP_PORT`: The TCP port that the BMC Remedy AR System server uses. The dispatcher is randomly assigned to an available port if you do not specify this option.
- `AR_SERVER_INFO_THREAD_LOG_FILE`: The name of the file to use if thread tracing is turned on (see `AR_SERVER_INFO_DEBUG_MODE`).
- `AR_SERVER_INFO_TWO_DIGIT_YEAR_CUTOFF`: An integer that indicates the two-digit year cutoff.
- `AR_SERVER_INFO_U_CACHE_CHANGE` (get only): The time of the last change to the user cache.
- `AR_SERVER_INFO_UNQUAL_QUERIES`: A flag that indicates whether the server allows unqualified queries. Unqualified queries are `ARGetListEntry` calls in which the `qualifier` parameter is either `NULL` or has an operation value of `0` (`AR_COND_OP_NONE`). These queries can cause performance problems, especially for large forms, because they return *all* entries for a given form. If not allowed, you can return all form entries by specifying a "dummy" qualification such as `1` = `1`. Valid values for this option are `1` (`TRUE`) and `0` (`FALSE`). The default value is `TRUE` (allow unqualified queries).
- `AR_SERVER_INFO_USE_CON_NAME_IN_STATS`: A integer that indicates whether to use the default server name or the server connect name in statistics entries. Valid values are `0` (default name) and `1` (connect name). The default value is `0`.
- `AR_SERVER_INFO_USE_ETC_PASSWD`: A flag that indicates whether the **/etc/passwd** file is used to validate users not registered with the BMC Remedy AR System (UNIX only). If so, users in **/etc/passwd** are considered valid users of the BMC Remedy AR System and are assigned to a group identified by the UNIX group ID. Valid values for this option are `1` (`TRUE`) and `0` (`FALSE`). The default value is `1` (use password file).
- `AR_SERVER_INFO_USER_CACHE_UTILS`: A flag that indicates whether the `arcache` and `arreload` utilities are enabled. Valid values for this option are `0` (disabled) and `1` (enabled).
- `AR_SERVER_INFO_USER_LOG_FILE`: The name of the file to use if user tracing is turned on (see `AR_SERVER_INFO_DEBUG_MODE`).

- AR_SERVER_INFO_VERSION (get only): The BMC Remedy AR System server version (character string).
- AR_SERVER_INFO_XREF_PASSWORDS: A flag that indicates whether the **/etc/passwd** file is cross-referenced if an BMC Remedy AR System user has no password (UNIX only). This option enables you to manage group membership and other support information by using the BMC Remedy AR System but still manage passwords using the **/etc/passwd** file. Valid values for this option are 1 (TRUE) and 0 (FALSE). The default value is 0 (blank passwords are not cross referenced).

See also

ARGetServerStatistics, ARSetServerInfo. See FreeAR for: FreeARServerInfoList, FreeARServerInfoRequestList, FreeARStatusList.

6.2.108 ARGetSessionConfiguration

Description

Retrieves a public API session variable.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetSessionConfiguration(
   ARControlStruct *control,
   unsigned int variableId,
   ARValueStruct *variableValue,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **sessionId** fields are required.

variableId

Identification for the variable type. Identification of the variable type:

`1:`	Maximum size for a single response (`AR_SESS_CHUNK_RESPONSE_SIZE`).
`2:`	Timeout for normal operations (`AR_SESS_TIMEOUT_NORMAL`).
`3:`	Timeout for long operations (`AR_SESS_TIMEOUT_LONG`).
`4:`	Timeout for extra long operations (`AR_SESS_TIMEOUT_XLONG`).
`5:`	Socket number to lock to (`AR_SESS_LOCK_TO_SOCKET_NUMBER`).
`6:`	Flag with `Boolean` value that indicates if the session is pooled (`AR_SESS_POOLED`).
`7:`	API program client type (`AR_SESS_CLIENT_TYPE`).
`8:`	Type of VUI view (`AR_SESS_VUI_TYPE`).
`9:`	Flag with `Boolean` value that indicates if the user would like to override the session from the previous IP (`AR_SESS_OVERRIDE_PREV_IP`).
`10:`	The name of the API command log (`AR_SESS_API_CMD_LOG`).
`11:`	The name of the API result log (`AR_SESS_API_RES_LOG`).
`12:`	Overlay group used at design time (`AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP`).
`13:`	Overlay group used at runtime (`AR_SESS_CONTROL_PROP_API_OVERLAYGROUP`).
`14:`	Granular overlay mode used by Get APIs. For extended properties, determines whether full lists or only extensions are retrieved. (`AR_SESS_CONTROL_PROP_GRANULAR_OVERLAY_RETRIEVE_MODE`).

variableValue

Configuration variable value:

- `AR_SESS_CHUNK_RESPONSE_SIZE` (integer): An integer value for the maximum data packet size returned from the server to the client in one transmission.
- `AR_SESS_TIMEOUT_NORMAL` (integer): An integer value for the client timeout value used in fast server operations.
- `AR_SESS_TIMEOUT_LONG` (integer): An integer value for the client timeout value used in list server operations.
- `AR_SESS_TIMEOUT_XLONG` (integer): An integer value for the client timeout value used in definition updating and in import/export operations.
- `AR_SESS_LOCK_TO_SOCKET_NUMBER` (integer): An integer value for the socket number that the client locks to for all server interaction.
- `AR_SESS_POOLED` (integer): An integer value for the flag that indicates if the session is pooled. Valid values for this option are `0` (`No`) and `1` (`Yes`).
- `AR_CLIENT_TYPE_` (integer): An integer value for the client type. For more information, see `AR_CLIENT_TYPE_` in the **ar.h** file.
  Range of values for user-defined, client-type variables:
  `AR_CLIENT_TYPE_END_OF_RESERVED_RANGE: >5000`.

- `AR_SESS_VUI_TYPE` (integer): An integer value for the type of VUI. For more information, see `AR_VUI_TYPE_` in the **ar.h** file.
- `AR_SESS_OVERRIDE_PREV_IP` (integer): An integer value for the flag that indicates if the session can be overridden. Valid values for this option are `0` (`No`) and `1` (`Yes`).
- `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` (character): A character value that indicates whether to use an object's real or resolved name in administrative and design-time operations (`ARCreate`, `ARDelete`, and `ARSet` calls) and whether to perform the operation only on objects in a specified overlay group. (Unmodified objects are members of all groups, so they are included in all operations regardless of the value of this variable.)
  Valid values for this option are the following strings:

• `None` *or* `0` — Use real names; perform operation on origin (overlaid and unmodified) objects.
• `overlayGroupID` — Use resolved names; perform operation on objects in the specified overlay group. (In AR System 7.6.04, you can only specify group ID `1`.)
• `-1` — (Base Development mode) Use real names; perform operation on origin (overlaid and unmodified) objects.
• `-2` — (Full Development mode) Use real names; perform operation on all objects (overlay, overlaid, unmodified, and custom).

- `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` (character): A character value that indicates whether to use an object's real or resolved name in runtime operations (`ARGet`, `ARGetList`, and `ARGetMultiple` calls) and whether to perform the operation only on custom and overlay objects in a specified overlay group. (Unmodified objects are members of all groups, so they are included in all operations regardless of the value of this variable.)
  Valid values for this option are the following strings:

• `None` *or* `0` — Use real names; perform operation on origin (overlaid and unmodified) objects.
• `overlayGroupID` — Use resolved names; perform operation on objects in the specified overlay group. (In AR System 7.6.04, you can only specify group ID `1`.)
• `-1` — (Base Development mode) Use real names; perform operation on origin (overlaid and unmodified) objects.
• `-2` — (Full Development mode) Use real names; perform operation on all objects (overlay, overlaid, unmodified, and custom).

- `AR_SESS_CONTROL_PROP_GRANULAR_OVERLAY_RETRIEVE_MODE (int)`: An integer value that defines what type of object definition is returned. Valid values for this option are:
  - `0` (the default): All Get<object> calls return the complete object definition.
  - `1:` For grains of overlays that can be extended (form lists, permissions, and indexes), Get<object> calls return only the values used to extend or overwrite the values retrieved from the origin object.
    For all other grains, the inherited or overwritten values are returned.
- `AR_SESS_API_CMD_LOG` (char): A character value for the name of the API client side logging command file.

- AR_SESS_API_RES_LOG (char): A character value for the name of the API client side logging result file.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetSessionConfiguration.

6.2.109 ARGetServerStatistics

Description

Retrieves the requested statistics for the specified server. The counts returned generally represent the number of occurrences since starting the server. If a statistic reaches the maximum value for a long integer, the system resets the counter and begins incrementing again.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARGetServerStatistics(
   ARControlStruct *control,
   ARServerInfoRequestList *requestList,
   ARServerInfoList *serverInfo,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

requestList

A list of one or more statistics to return (see the Statistics Options section that follows).

Return values

serverInfo

The statistics retrieved from the server (see the Statistic Options that follows). The system returns `NULL` (`AR_DATA_TYPE_NULL`) for statistics not retrieved due to error.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

Statistics options

- `AR_SERVER_STAT_API_REQUESTS` (integer): The total number of API requests received.
- `AR_SERVER_STAT_API_TIME` (integer): The total processor time (in 1/100 of a second) spent performing API function calls.
- `AR_SERVER_STAT_BAD_PASSWORD` (integer): The total number of times an incorrect password was specified during login.
- `AR_SERVER_STAT_CACHE_TIME` (integer): The total processor time (in 1/100 of a second) spent loading the internal cache to improve performance.
- `AR_SERVER_STAT_CPU` (integer): The total CPU time (in 1/100 of a second) used by the server.
- `AR_SERVER_STAT_CREATE_E_COUNT` (integer): The total number of calls made to the `ARCreateEntry` function.
- `AR_SERVER_STAT_CREATE_E_TIME` (integer): The total processor time (in 1/100 of a second) spent performing the `ARCreateEntry` function.
- `AR_SERVER_STAT_CURRENT_USERS` (integer): The total number of users currently accessing the system.
- `AR_SERVER_STAT_DELETE_E_COUNT` (integer): The total number of calls made to the `ARDeleteEntry` function.
- `AR_SERVER_STAT_DELETE_E_TIME` (integer): The total processor time (in 1/100 of a second) spent performing the `ARDeleteEntry` function.
- `AR_SERVER_STAT_ENTRY_TIME` (integer): The total processor time (in 1/100 of a second) spent performing API function calls that manipulate entries.
- `AR_SERVER_STAT_ESCL_DISABLE` (integer): The total number of escalations that were evaluated but skipped because they were marked disabled.
- `AR_SERVER_STAT_ESCL_FAILED` (integer): The total number of escalations that were skipped (qualification criteria not met).
- `AR_SERVER_STAT_ESCL_FIELDP` (integer): The total number of Push Fields escalation actions performed.
- `AR_SERVER_STAT_ESCL_FIELDS` (integer): The total number of Set Fields escalation actions performed.

- `AR_SERVER_STAT_ESCL_FIELDS_FLTAPI` (integer): The number of escalations that modify fields through the API.
- `AR_SERVER_STAT_ESCL_FIELDS_PROCESS` (integer): The number of escalations that modify fields.
- `AR_SERVER_STAT_ESCL_FIELDS_SQL` (integer): The number of escalations that modify SQL fields.
- `AR_SERVER_STAT_ESCL_LOG` (integer): The total number of log escalation actions performed.
- `AR_SERVER_STAT_ESCL_NOTIFY` (integer): The total number of notify escalation actions performed.
- `AR_SERVER_STAT_ESCL_PASSED` (integer): The total number of escalations that were executed (qualification criteria met).
- `AR_SERVER_STAT_ESCL_PROCESS` (integer): The total number of run process escalation actions performed.
- `AR_SERVER_STAT_ESCL_SQL` (integer): The total number of direct SQL escalation actions performed.
- `AR_SERVER_STAT_ESCL_TIME` (integer): The total processor time (in 1/100 of a second) spent checking and processing escalations.
- `AR_SERVER_STAT_E_STATS_COUNT` (integer): The total number of calls made to the `ARGetEntryStatistics` function.
- `AR_SERVER_STAT_E_STATS_TIME` (integer): The total processor time (in 1/100 of a second) spent performing the `ARGetEntryStatistics` function.
- `AR_SERVER_STAT_FILTER_CALL_GUIDE` (integer): The number of filter guide actions that call another process.
- `AR_SERVER_STAT_FILTER_DISABLE` (integer): The total number of filters that were evaluated but skipped because they were marked disabled.
- `AR_SERVER_STAT_FILTER_EXIT_GUIDE` (integer): The number of filter guide actions that are terminated.
- `AR_SERVER_STAT_FILTER_FAILED` (integer): The total number of filters that were skipped (qualification criteria not met).
- `AR_SERVER_STAT_FILTER_FIELDP` (integer): The total number of Push Fields filter actions performed.
- `AR_SERVER_STAT_FILTER_FIELDS` (integer): The total number of Set Fields filter actions performed.
- `AR_SERVER_STAT_FILTER_FIELDS_SQL` (integer): The number of filters that modify SQL fields.
- `AR_SERVER_STAT_FILTER_FIELDS_FLTAPI70` (integer): The number of filter actions that modify fields through the API.
- `AR_SERVER_STAT_FILTER_FIELDS_PROCESS` (integer): The number of filter actions that modify fields.
- `AR_SERVER_STAT_FILTER_GOTO_ACTION` (integer): The number of filter actions that branch to another process.
- `AR_SERVER_STAT_FILTER_LOG` (integer): The total number of log filter actions performed.
- `AR_SERVER_STAT_FILTER_MESSAGE` (integer): The total number of message filter actions performed.
- `AR_SERVER_STAT_FILTER_NOTIFY` (integer): The total number of notify filter actions performed.
- `AR_SERVER_STAT_FILTER_PASSED` (integer): The total number of filters that were executed (qualification criteria met).

- AR_SERVER_STAT_FILTER_PROCESS (integer): The total number of run process filter actions performed.
- AR_SERVER_STAT_FILTER_SQL (integer): The total number of direct SQL filter actions performed.
- AR_SERVER_STAT_FILTER_TIME (integer): The total processor time (in 1/100 of a second) spent checking and processing filters.
- AR_SERVER_STAT_FTS_SRCH_COUNT (integer): The total number of FTS operations performed.
- AR_SERVER_STAT_FTS_SRCH_TIME (integer): The total processor time (in 1/100 of a second) spent performing FTS operations.
- AR_SERVER_STAT_FULL_FIXED (integer): The total number of connected users with fixed FTS licenses.
- AR_SERVER_STAT_FULL_FLOATING (integer): The total number of connected users with floating FTS licenses.
- AR_SERVER_STAT_FULL_NONE (integer): The total number of connected users with no FTS license.
- AR_SERVER_STAT_GET_E_COUNT (integer): The total number of calls made to the ARGetEntry function.
- AR_SERVER_STAT_GET_E_TIME (integer): The total processor time (in 1/100 of a second) spent performing the ARGetEntry function.
- AR_SERVER_STAT_GETLIST_E_COUNT (integer): The total number of calls made to the ARGetListEntry function.
- AR_SERVER_STAT_GETLIST_E_TIME (integer): The total processor time (in 1/100 of a second) spent performing the ARGetListEntry function.
- AR_SERVER_STAT_IDLE_TIME (integer): The total idle time (in 1/100 of a second) of the server threads when the threads are not processing any requests. The idle time for each thread is determined each time it begins to process a new request. The idle time accumulated after the most recent activity in each thread is not included in this value. Therefore, the total idle time value can grow in an unexpected manner depending on the frequency of requests being handled by the server threads and how those requests are distributed amongst various threads. The accuracy of the idle time value will increase as the level of activity on the server increases and all threads are handling requests.
- AR_SERVER_STAT_MAX_CACHES (integer): The maximum cache count.
- AR_SERVER_STAT_MERGE_E_COUNT (integer): The total number of calls made to the ARMergeEntry function.
- AR_SERVER_STAT_MERGE_E_TIME (integer): The total processor time (in 1/100 of a second) spent performing the ARMergeEntry function.
- AR_SERVER_STAT_NET_RESP_TIME (integer): The total time (in 1/100 of a second) spent on the network responding to the client.
- AR_SERVER_STAT_NO_FULL_TOKEN (integer): The total number of times a user tried to connect and no floating FTS token was available.
- AR_SERVER_STAT_NO_WRITE_TOKEN (integer): The total number of times a user tried to connect and no floating write token was available.
- AR_SERVER_STAT_NUM_CACHES (integer): The number of current caches.
- AR_SERVER_STAT_NUM_THREADS (integer): The number of threads in a particular queue.
- AR_SERVER_STAT_NUMBER_BLOCKED (integer): The total number of blocked processes. This statistic, in conjunction with the AR_SERVER_STAT_TIMES_BLOCKED statistic, enables you to

determine the approximate number of processes being blocked per instance. For example, if `AR_SERVER_STAT_TIMES_BLOCKED` is three and `AR_SERVER_STAT_NUMBER_BLOCKED` is six, the six blocked processes could be distributed in the following possible ways:

  - Two blocked processes in each of the three instances (2+2+2 = 6).
  - Three blocked processes in one instance, two blocked processes in one instance, and one blocked process in one instance (3+2+1 = 6).
  - Four blocked processes in one instance and one blocked process in both of the other two instances (4+1+1 = 6). In this example, the blocked processes could not be distributed as five in one instance, one in one instance, and none in one instance because, by definition, the number of instances represents those times when at least one process is blocked.

- `AR_SERVER_STAT_OTHER_TIME` (integer): The total processor time (in 1/100 of a second) spent performing API function calls that do not manipulate entries or restructure the database.
- `AR_SERVER_STAT_RESTRUCT_TIME` (integer): The total processor time (in 1/100 of a second) spent performing API function calls that restructure the database.
- `AR_SERVER_STAT_WRITE_RESTRICTED_READ` (integer): The total number of connected users with restricted read licenses.
- `AR_SERVER_STAT_SET_E_COUNT` (integer): The total number of calls made to the `ARSetEntry` function.
- `AR_SERVER_STAT_SET_E_TIME` (integer): The total processor time (in 1/100 of a second) spent performing the `ARSetEntry` function.
- `AR_SERVER_STAT_SINCE_START` (integer): The number of seconds (in 1/100 of a second) since the server was started. That is, the total real time that the server process has been running.
- `AR_SERVER_STAT_SQL_DB_COUNT` (integer): The total number of SQL commands sent to the database.
- `AR_SERVER_STAT_SQL_DB_TIME` (integer): The total processor time (in 1/100 of a second) spent performing SQL commands.
- `AR_SERVER_STAT_START_TIME` (time stamp): The UNIX time at which this server was started.
- `AR_SERVER_STAT_TIMES_BLOCKED` (integer): The total number of instances in which at least one process was blocked by the current process. This statistic, in conjunction with the `AR_SERVER_STAT_NUMBER_BLOCKED` statistic, enables you to determine the approximate number of processes being blocked per instance.
- `AR_SERVER_STAT_WRITE_FIXED` (integer): The total number of connected users with fixed write licenses.
- `AR_SERVER_STAT_WRITE_FLOATING` (integer): The total number of connected users with floating write licenses.
- `AR_SERVER_STAT_WRITE_READ` (integer): The total number of connected users with no write license.

See also

ARGetServerInfo. See FreeAR for: `FreeARServerInfoList`, `FreeARServerInfoRequestList`, `FreeARStatusList`.

6.2.110 ARGetSupportFile

Description

Retrieves a file supporting external reports.

Privileges

Any user who has access to the specified object.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetSupportFile(
   ARControlStruct *control,
   unsigned int  fileType,
   ARNameType name,
   ARInternalId id2,
   ARInternalId fileId,
   FILE *filePtr,
   ARTimestamp *timeStamp,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

fileType

The numerical value for the type of file, and the type of object the file is associated with. Specify `1` (`AR_SUPPORT_FILE_EXTERNAL_REPORT`) for an external report file associated with an active link.

name

The name of the object the file is associated with, usually a form.

id2

The ID of the field or VUI, if the object is a form. If the object is not a form, set this parameter to `0`.

fileId

The unique identifier of a file within its object.

filePtr

A pointer to a file from which the support file contents are retrieved. Specify `NULL` for this parameter if you do not want to retrieve the file contents. If you are using Windows, you must open the file in binary mode.

Return values

timeStamp

A time stamp that specifies the last change to the field. Specify `NULL` for this parameter if you do not want to retrieve this value.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent, ARCreateSupportFile, ARDeleteActiveLink, ARDeleteSupportFile, ARGetActiveLink, ARGetListSupportFile, ARSetActiveLink, ARSetSupportFile.

6.2.111 ARGetTextForErrorMessage

Description

Retrieves the message text for the specified error from the local catalog (in the local language). The length of the text is limited by `AR_MAX_MESSAGE_SIZE` (255 bytes).

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARGetTextForErrorMessage(
   char *ARGetTextForErrorMessage (
   int msgId)
```

Input arguments

msgId

The error number whose message text you want to retrieve. This number is provided in the `messageNum` member of `ARStatusStruct`.

Return values

return

The function return value is a pointer to the retrieved message text. You must free this memory after calling this function.

6.2.112 ARGetVUI

Description

Retrieves information about the form view (VUI) with the indicated ID from the specified server.

Privileges

This operation can be performed by users with access permission for the form with which the VUI is associated.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARGetVUI(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalId vuiId,
   ARNameType vuiName,
   ARLocaleType locale,
   unsigned int *vuiType,
   ARPropList *dPropList,
   char **helpText,
   ARTimestamp *timestamp,
   ARAccessNameType owner,
   ARAccessNameType lastChanged,
   char **changeDiary,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

To specify whether to search for an object's real or resolved name, use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration). For calls that use field or view IDs, the search is performed on the real or resolved name of the associated form.

If you do not use the `AR_SESS_CONTROL_PROP_API_OVERLAYGROUP` variable, the BMC Remedy AR System server uses the default overlay group at run time. For information about configuring the default overlay group, see the description for `Overlay-mode` in ar.cfg or ar.conf options N-R.

schema

The name of the form containing the VUI to retrieve.

vuiId

The internal ID of the VUI to retrieve.

Return values

vuiName

The name of the VUI. Specify `NULL` for this parameter if you do not want to retrieve the VUI name.

locale

The locale of the VUI. Specify `NULL` for this parameter if you do not want to retrieve the locale.

vuiType

The type of VUI. Specify `NULL` for this parameter if you do not want to retrieve the VUI type.

`0:`	No VUI type (`AR_VUI_TYPE_NONE`).
`1:`	Windows type (`AR_VUI_TYPE_WINDOWS`) * - fields in BMC Remedy User.
`2:`	Web relative type (`AR_VUI_TYPE_WEB`) - fields in view can be adjusted.
`3:`	Web absolute type (`AR_VUI_TYPE_WEB_ABS_POS`) - fields in view are fixed.

Field marked with an asterisk (*) is available for legacy environments that use BMC Remedy User, which is no longer shipped with BMC Remedy AR System.

dPropList

A list of zero or more display properties associated with the VUI. See ARCreateVUI for a description of the possible values. The system returns `0` (`AR_DPROP_NONE`) if the VUI has no display properties. Specify `NULL` for this parameter if you do not want to retrieve this list.

helpText

The help text associated with the VUI. Specify `NULL` for this parameter if you do not want to retrieve the help text. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

timestamp

A time stamp that specifies the last change to the VUI. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owning user for the VUI. Specify `NULL` for this parameter if you do not want to retrieve the owner.

lastChanged

The user who made the last change to the VUI. Specify `NULL` for this parameter if you do not want to retrieve this value.

changeDiary

The change diary associated with the VUI. Use `ARDecodeDiary` to parse the change diary into user, time stamp, and text components. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to retrieve the change diary. This value can be large, so do not retrieve it if the program does not need it, for example, if the program is only checking it an object exists.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties list is returned. See Server object properties.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateVUI, ARDecodeDiary, ARDeleteVUI, ARGetListVUI, ARGetMultipleVUIs, ARSetVUI. See FreeAR for: `FreeARPropList`, `FreeARStatusList`.

6.2.113 ARImport

Description

Imports the indicated structure definitions to the specified server. Use this function to copy structure definitions from one BMC Remedy AR System server to another (see ARExport).

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARImport(
   ARControlStruct *control,
   ARStructItemList *structItems,
   char *importBuf,
   unsigned int importOption,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

The runtime group specified in the control record does not affect import operations. Instead, that function uses the overlay group specified in the object property list in the imported object definition file. For information about runtime groups and the overlay group setting, see ARSetSessionConfiguration.

structItems

A list of zero or more structure items to import (identified by type and name). Specify `NULL` for this parameter to import all structures in the import buffer.

1:	Form (includes all associated views) (`AR_STRUCT_ITEM_SCHEMA`).
5:	Filter (`AR_STRUCT_ITEM_FILTER`).
6:	Active link (`AR_STRUCT_ITEM_ACTIVE_LINK`).
8:	Character menu (`AR_STRUCT_ITEM_CHAR_MENU`).
9:	Escalation (`AR_STRUCT_ITEM_ESCALATION`).
10:	Distributed mapping (`AR_STRUCT_ITEM_DIST_MAP`).
12:	Container (`AR_STRUCT_ITEM_CONTAINER`).
14:	VUI (`AR_STRUCT_ITEM_VUI`).
16:	Application (`AR_STRUCT_ITEM_APP`)

30:	Form data (AR_STRUCT_ITEM_SCHEMA_DATA)

Note

You must specify AR_STRUCT_ITEM_SCHEMA to import a form. The three partial form types defined in **ar.h** (AR_STRUCT_ITEM_SCHEMA_DEFN, AR_STRUCT_ITEM_SCHEMA_VIEW, and AR_STRUCT_ITEM_SCHEMA_MAIL) do not contain complete form definitions and are used only for caching purposes.

Note

When you import an overlay that has inherited properties, those properties are inherited from the copy of the origin object that is on the server. If you import the origin object with the overlay, then the overlay will inherit properties from the imported object. If not, then the overlay will inherit properties from the object already on the server.

importBuf

A buffer that contains the definition text for the items specified for the `structItems` parameter. This buffer can contain other structure definitions, but the system imports only the specified items.

importOption

A value that indicates whether to replace objects being imported if they already exist and how to replace them. There are two sets of import options. Always set one of the basic options to indicate whether to replace objects. Set zero or more advanced options to further configure how `ARImport` handles inconsistencies between the objects on the server and the object definitions in the import buffer.

Basic options:

0:	Create a new objects and generate an error if an object already exists. (AR_IMPORT_OPT_CREATE)
1:	Overwrite an existing object. (AR_IMPORT_OPT_OVERWRITE)

Advanced options (add the values to enable more than one option):

16:	Check for conflicting data types and report an error. (AR_IMPORT_OPT_HANDLE_CONFLICT_ERROR).
32:	Check for conflicting data types and overwrite the existing object with the imported object. (AR_IMPORT_OPT_HANDLE_CONFLICT_OVERWRITE)
48:	Check for conflicting data types and do not change the existing object if there is a conflict. (AR_IMPORT_OPT_HANDLE_CONFLICT_NO_ACTION).
64:	Do not delete fields on the existing schema that are not in the imported schema. (AR_IMPORT_OPT_NOT_DELETE_FIELD).

`128`:	Check for duplicate entries and report an error. (`AR_IMPORT_OPT_DATA_REJECT_FOR_DUP`)
`256`:	Generate a new ID for a duplicate entry. (`AR_IMPORT_OPT_DATA_NEWID_FOR_DUP`)
`512`:	Overwrite existing entry with an imported duplicate. (`AR_IMPORT_OPT_DATA_OVERWRITE_FOR_DUP`)
`1024`:	Merge an existing entry with an imported duplicate. (`AR_IMPORT_OPT_DATA_MERGE_FOR_DUP`)
`2048`:	Generate a new ID for each entry. (`AR_IMPORT_OPT_DATA_NEWID_FOR_ALL`)
`4096`:	Do not delete VUIs of the existing schema that are not in the imported schema. (`AR_IMPORT_OPT_NOT_DELETE_VUI`).
`8192`:	Preserve the permissions of existing objects. (`AR_IMPORT_OPT_NOT_OVERWRITE_PERMISSION`)
`16384`:	Preserve all existing workflow attributes except the form list, the list of forms the workflow is attached to. (`AR_IMPORT_OPT_WORKFLOW_PRESERVE_DEFN`)
`32768`:	Add the forms in the imported workflow form list to the workflow form list of an existing schema instead of replacing the workflow form list of an existing schema with the imported one. (`AR_IMPORT_OPT_WORKFLOW_MERGE_ATTACHLIST`)
`65536`:	Preserve the change history of an existing object instead of replacing it with the change history of the imported object. (`AR_IMPORT_OPT_PRESERVE_HISTORY`)
`2097152`:	Overwrite all workflow attributes, but preserve the existing form list. (`AR_IMPORT_OPT_WORKFLOW_PRESERVE_ATTACHLIST`)
`4194304`:	Remove the forms included in the imported workflow form list from the workflow form list of the existing schema, but preserve the existing workflow attributes. (`AR_IMPORT_OPT_WORKFLOW_REMOVE_ATTACHLIST`)
`8388608`:	Overwrite full text option on existing field. (`AR_IMPORT_OPT_OVERWRITE_FULL_TEXT_OPTION`)
`16777216`:	Overwrite field display properties with properties from definition. (`AR_IMPORT_OPT_OVERWRITE_DISP_PROPS`)

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARExport, ARExportToFile. See FreeAR for: FreeARStatusList, FreeARStructItemList.

6.2.114 ARImportLicense

Description

Imports the contents of a license file (**arsystem.lic** or a similarly formatted file) from a buffer into the Add or Remove Licenses form. Based on the value of the `importOption` argument, performs the import in one of these ways:

- Clears the contents of the form and replaces it with the contents of the file. Multiple entries in the file for the same license type are merged into one entry in the form.
- Merges the contents of the file with the contents of the form as follows:
    - When the form and the file contain a matching license type, the entries for that type are *not imported* from the file.

For example, if the form has an entry for five AR User Fixed licenses and the file has an entry for three AR User Fixed licenses and another entry for four AR User Fixed licenses, the form retains its entry, and neither entry is imported from the file.

- When the file contains multiple entries for a particular license type and the form contains no matching entry, all entries in the file are imported into the form and merged into one entry.
  For example, if the form has no entries for AR User Fixed licenses and the file has an entry for three AR User Fixed licenses plus another entry for four AR User Fixed licenses, the two entries in the file are merged into one entry in the form. The new entry in the form is for seven AR User Fixed licenses.

When called, the server validates that the buffer contains the text from a validly formatted license file, but it does not validate checksums or encryption.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARImportLicense(
   ARControlStruct *control,
   char *importBuf,
   unsigned int importOption,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

importBuf

The text from a license file.

importOption

Indicates whether the license data is to be merged or overwritten.

`0:`	Merge the file and form data, summing the matching license types (`AR_LICENSE_IMPORT_APPEND`).
`1:`	Overwrite the data in the form (`AR_LICENSE_IMPORT_OVERWRITE`).

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARExportLicense.

6.2.115 ARInitialization

Description

Performs server- and network-specific initialization operations for each BMC Remedy AR System session. All API programs that interact with the BMC Remedy AR System must call this function before calling any other BMC Remedy AR System API functions. Your program can call this function again to create additional sessions.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARInitialization(
   ARControlStruct *control,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. The `control` parameter must be the first input argument. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required. The **sessionId** field, which you must supply in the control record for most other functions, is returned by this function.

To use Unicode (UTF-8) as the client character set, set the **ARControlStruct localInfo.charSet** field as follows:

```
memset(&control, '\0', sizeof(control));
strcpy(control.localeInfo.charSet, "UTF-8");
...
if (ARInitialization(&control, &status) >= AR_RETURN_ERROR)
```

Pass an empty string as **localeInfo.charSet** to get the normal API behavior. That is, the client character set is the character set implied by the client's locale.

The other fields of `ARControlStruct` may safely be null-valued, and `ARInitialization` automatically fills them in.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

See FreeAR for: `FreeARStatusList`.

6.2.116 ARJulianDateToDate

Description

Converts a Julian date to a year, month, and day value. The Julian date is the number of days since noon, Universal Time, on January 1, 4713 BCE (on the Julian calendar). The changeover from the Julian calendar to the Gregorian calendar occurred in October, 1582. The Julian calendar is used for dates on or before October 4, 1582. The Gregorian calendar is used for dates on or after October 15, 1582.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARJulianDateToDate(
   ARControlStruct *control,
   ARDateStruct *date,
   int jd,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

jd

The Julian date value to convert.

Return values

date

The resulting date structure holding the year, month, and day value.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDateToJulianDate.

6.2.117 ARLoadARQualifierStruct

Description

Loads the specified qualification string (if valid for the form) into an `ARQualifierStruct` structure. This function simplifies the process of specifying qualifications in the required format.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARLoadARQualifierStruct(
   ARControlStruct *control,
   ARNameType schema,
   ARNameType displayTag,
   char *qualString,
   ARQualifierStruct *qualifier,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form to which the qualification applies.

displayTag

The name of the form view (VUI) to use for resolving field names. If the specified view does not exist or does not contain a field specified in the qualification string (or you specify **NULL** for this parameter), the system uses the field label or field ID in the default view. If the field is not available on the view, field label is used. If the view name is not provided, the default view is used.

qualString

A character string containing the qualification to load (following the syntax rules for entering qualifications in BMC Remedy Mid Tier).

Return values

qualifier

An `ARQualifierStruct` structure that contains the specified qualification. Use `FreeARQualifierStruct` to recursively free the allocated memory associated with this structure as soon as you no longer need it.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetEntryStatistics, ARGetListEntry. See FreeAR for: `FreeARQualifierStruct`, `FreeARStatusList`.

6.2.118 ARMergeEntry

Description

Merges an existing database entry into the indicated form on the specified server.

Note

When operating on join forms, this function simply triggers workflow. You must create workflow to propagate data to the join form's underlying forms. For more information, see Performing distributed operations on join forms.

Privileges

The system merges data based on the access privileges of the user that you specify for the `control` parameter and the **createMode** setting for each field (see ARCreateField). User permissions are verified for each specified field. The system generates an error if the user does not have write permission for a field or a field does not exist.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARMergeEntry(
   ARControlStruct *control,
   ARNameType schema,
   ARFieldValueList *fieldList,
   unsigned int mergeType,
   ARQualifierStruct *query,
   unsigned int multimatchOption,
   AREntryIdType *entryId,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form to merge the entry into.

fieldList

A list of one or more field/value pairs (specified in any order) that identifies the data for the new entry. You must specify values for all required fields that do not have defined defaults. Values must be of the data type defined for the field or have a data type of `0` (`AR_DATA_TYPE_NULL`). `NULL` values can be specified only for optional fields, and assigning `NULL` overrides any defined field default. An error is generated if a field does not exist or the user specified by the **control** parameter does not have write permission for a field.

Note

You must specify a formatted diary string (such as that returned by `ARGetEntry`) for any diary fields that you want to merge. In addition, unlike creating a new entry, you can specify values for the Entry ID, Create Date, Last Modified By, Modified Date, and Status History fields when merging an existing entry.

mergeType

Note

This argument is not applicable to join forms.

A value that indicates the action to take if `fieldList` includes the Entry ID field and the ID specified already exists in the target form. This parameter is ignored if you do not specify the Entry ID field or the ID specified does not conflict with existing entry IDs.

`1:`	Generate an error (`AR_MERGE_ENTRY_DUP_ERROR`).
`2:`	Create a new entry with a new ID (`AR_MERGE_ENTRY_DUP_NEW_ID`).
`3:`	Delete the existing entry and create a new one in its place (`AR_MERGE_ENTRY_DUP_OVERWRITE`).
`4:`	Update the fields specified in `fieldList` in the existing entry (`AR_MERGE_ENTRY_DUP_MERGE`).
`5:`	Always generate a new entry ID for the record, even if there is no conflicting entry ID (`AR_MERGE_ENTRY_GEN_NEW_ID`).

To omit some field validation steps, add the appropriate increments to the merge type.

`1024:`	Allow NULL in required fields (not applicable to the Submitter, Status, or Short-Description core fields) (`AR_MERGE_NO_REQUIRED_INCREMENT`).
`2048:`	Skip field pattern checking (including $MENU$) (`AR_MERGE_NO_PATTERNS_INCREMENT`).
`4096:`	Ignore filter processing (`AR_MERGE_NO_WORKFLOW_FIRED`).

query

A query that determines the set of entries to retrieve. The qualification can include one or more fields and any combination of conditional, relational, and arithmetic (numeric data types only) operations. The system generates an error if the user does not have read permission for a field or a field does not exist.

multimatchOption

A flag that determines the action to be taken when there are multiple matching entries for a given query. This parameter is valid only if a query is specified, else the server ignores the parameter.

Valid values are:

- `0`: Error on multi-match (`AR_MERGE_ENTRY_MULT_MATCH_ERROR`)
- `1`: Use first match for the MergeEntry operation (`AR_MERGE_ENTRY_MULT_MATCH_USE_FIRST`)

Return values

entryId

The ID of the merged entry. If you do not specify the Entry ID field in `fieldList`, the system generates and returns a new ID. If you do specify the Entry ID field and the ID specified is unique in the target form, the system returns that ID. If the ID specified is not unique and you specify `AR_MERGE_ENTRY_DUP_NEW_ID` for the `mergeType` parameter, the system generates and returns a new ID.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEntry. See FreeAR for: `FreeARFieldValueList`, `FreeARStatusList`.

6.2.119 ARRegisterForAlerts

Description

Registers the specified user with the BMC Remedy AR System server to receive alerts.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARRegisterForAlerts(
   ARControlStruct *control,
   int clientPort,
   unsigned int registrationFlags,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

clientPort

The client port number.

registrationFlags

This value is reserved for future use and must be set to zero.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDeregisterForAlerts, ARCreateAlertEvent, ARGetAlertCount. See FreeAR for: `FreeARStatusList`.

6.2.120 ARRunEscalation

Description

Runs the specified escalation immediately.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

ARControlStruct *control,
   ARNameType escalationName,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

escalationName

The name of the escalation to run.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.121 ARServiceEntry

Description

Takes an entry ID with a list of input field values, executes filter objects, and returns a list of output fields without writing the specified entry to the database. Thus, it eliminates the `ARSetEntry` call, `ARGetEntry` call, the `ARDeleteEntry` call, and several filters.

This function can work with an BMC Remedy AR System web service to obtain external services, or with a Set Fields filter action to consume an internal BMC Remedy AR System service.

Privileges

The permissions in workflow use system permissions, so the `ARServiceEntry` function is not directly involved in modifying the current entry. Therefore, no issue exists with permissions on the *input* fields. Any field in the input list can be used in the workflow.

For the *output* fields, the system returns data based on the access privileges of the user that you specify for the `control` parameter. User permissions are verified for each specified field. The system returns values for accessible fields, and it returns warning messages for fields the user cannot access.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"
```

```
int ARServiceEntry(
   ARControlStruct *control,
   ARNameType formName,
   AREntryIdList *entryId,
   ARFieldValueList *inputFieldList,
   ARInternalIdList *idList,
   ARFieldValueList *outputFieldList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

formName

The name of the form where the service is performed.

entryId

The ID of the entry to be used in the transaction. The function does not update the entry; it simply uses it to get values (which are not defined after the filter runs).

If an output mapping's field is undefined and an entry ID is specified, a database call is required to retrieve the field value from the database. If entry ID is *not* present, database calls are not required, and the undefined field's value is `NULL`. (An undefined field value is one that is neither present in inputFieldList nor in the transaction.)

Note

The system identifies entries in join forms by concatenating the entry IDs from the member forms. As a result, an entry ID can consist of one or more values of type `AREntryIdType` and, therefore, is represented by the `AREntryIdList` structure.

inputFieldList

A list of one or more field/value pairs (specified in any order) that identifies the new data for the entry. Values must be of the data type defined for the field or have a data type of `0` (`AR_DATA_TYPE_NULL`).

idList

A list of zero or more internal IDs that specify the fields to retrieve. Specify `NULL` for this parameter (or zero fields) to retrieve all (accessible) fields. Specify `NULL` for both this parameter and the `outputFieldList` parameter if you do not want to retrieve any fields. To minimize network traffic, specify only the fields that you need if you do not require the data for all fields. If an attachment field is specified in the list, only its name, size, and compressed size are returned. Use `ARGetEntryBLOB` to retrieve the contents of the attachment.

Return values

outputFieldList

A list of zero or more field/value pairs that identifies the data for the specified entry. The fields are returned in the order specified by `idList`. If the user does not have permission for a specified field or the field does not exist, the system does not return a value for the field/value pair. Specify `NULL` for this parameter if you do not want to retrieve any field data.

status

A list of zero or more notes, warnings, or errors generated from a call to this function.

6.2.122 ARSetActiveLink

Description

Updates the active link with the indicated name on the specified server. The changes are added to the server immediately and returned to users who request information about active links. Because active links operate on clients, individual clients do not receive the updated definition until they reconnect to the form (thus reloading the form from the server).

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetActiveLink(
   ARControlStruct *control,
   ARNameType name,
   ARNameType newName,
   unsigned int *order,
   ARWorkflowConnectStruct *workflowConnect,
   ARInternalIdList *groupList,
   unsigned int *executeMask,
   ARInternalId *controlField,
   ARInternalId *focusField,
   unsigned int *enable,
   ARQualifierStruct *query,
   ARActiveLinkActionList *actionList,
   ARActiveLinkActionList *elseList,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   unsigned int *errorActlinkOptions,
   ARNameType errorActlinkName,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

name

The name of the active link to update.

newName

The new name for the active link. The names of all active links on a given server must be unique. Specify `NULL` for this parameter if you do not want to change the name of the active link.

order

A value between `0` and `1000` (inclusive) that determines the active link execution order. When multiple active links are associated with a form, the value associated with each active link determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to change the order.

workflowConnect

The list of form names the active link is linked to. The active link must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to change the list of forms. If the object is locked, you can append a form name to the list but you cannot remove a form name from the list.

groupList

A list of zero or more groups who can access this active link. Users can execute an active link if they belong to a group that has access to it. Specify an empty group list to define an active link accessible by users with administrator capability. Specify group ID `0` (Public) to provides access to all users. The group list that you specify replaces all existing group permissions. Specify `NULL` for this parameter if you do not want to change the group list.

executeMask

A bitmask that indicates the form operations that trigger the active link.

Bit 0:	Execute the active link when a user selects a button, toolbar button, or menu item specified by the controlField parameter (`AR_EXECUTE_ON_BUTTON`).
Bit 1:	Execute the active link when a user presses Return in the field specified by the focusField parameter (`AR_EXECUTE_ON_RETURN`).
Bit 2:	Execute the active link when a user submits an entry (*before* data is sent to the BMC Remedy AR System server) (`AR_EXECUTE_ON_SUBMIT`).
Bit 3:	Execute the active link when a user modifies an individual entry (*before* data is sent to the BMC Remedy AR System server) (`AR_EXECUTE_ON_MODIFY`).
Bit 4:	Execute the active link when a user displays an entry (*after* data is retrieved from the BMC Remedy AR System server) (`AR_EXECUTE_ON_DISPLAY`).
Bit 7:	Execute the active link when a user selects an item from a character menu associated with the field specified by the `focusField` parameter or selects a row in the table field specified by the `focusField` parameter (`AR_EXECUTE_ON_MENU_CHOICE`).
Bit 8:	Execute the active link when a user moves out of a field (either manually or though workflow) (`AR_EXECUTE_ON_LOSE_FOCUS`).
Bit 9:	Execute the active link when a user sets default values (either manually or with preference settings) (`AR_EXECUTE_ON_SET_DEFAULT`).
Bit 10:	Execute the active link when a user retrieves one or more entries (*before* the query is sent to the BMC Remedy AR System server) (`AR_EXECUTE_ON_QUERY`).
Bit 11:	Execute the active link when a user modifies an individual entry (*after* data is committed to the database) (`AR_EXECUTE_ON_AFTER_MODIFY`).
Bit 12:	Execute the active link when a user submits an entry (*after* data is committed to the database) (`AR_EXECUTE_ON_AFTER_SUBMIT`).
Bit 13:	Execute the active link when a user moves into a field (either manually or through workflow) (`AR_EXECUTE_ON_GAIN_FOCUS`).
Bit 14:	Execute the active link when a user opens any form window (`AR_EXECUTE_ON_WINDOW_OPEN`).
Bit 15:	Execute the active link when a user closes any form window (`AR_EXECUTE_ON_WINDOW_CLOSE`).

Bit 16:	Execute the active link when a user moves off an entry (that is, when a record is removed from the server) (AR_EXECUTE_ON_UNDISPLAY).
Bit 17:	Execute the active link when a user performs a Copy to Submit operation (AR_EXECUTE_ON_COPY_SUBMIT).

Specify NULL for this parameter if you do not want to change the execute mask.

controlField

The ID of the field that represents the button, toolbar button, or menu item associated with executing the active link. The AR_EXECUTE_ON_BUTTON condition (see executeMask) is ignored unless you specify this parameter. This parameter is ignored if you do not specify AR_EXECUTE_ON_BUTTON. Specify NULL for this parameter if you do not want to change the control field.

focusField

The ID of the field associated with executing the active link by pressing Return or selecting a character menu item. The AR_EXECUTE_ON_RETURN or AR_EXECUTE_ON_MENU_CHOICE conditions (see executeMask) are ignored unless you specify this parameter (you must create another active link to specify a different field for each condition). This parameter is ignored if you do not specify either condition. Specify NULL for this parameter if you do not want to change the focus field.

enable

A flag to enable or disable this active link. A value of 0 disables the active link, causing it to be invisible to the user and unavailable for use. A value of 1 enables the active link, causing it to be visible and available for use. Specify NULL for this parameter if you do not want to change this flag.

query

A qualification that determines whether the active link is executed. Assign an operation value of 0 (AR_COND_OP_NONE) to execute the active link unconditionally. Specify NULL for this parameter if you do not want to change the query.

actionList

The set of actions performed if the condition defined by the query parameter is satisfied. You can specify from 1 to 25 actions in this list (limited by AR_MAX_ACTIONS). Specify NULL for this parameter if you do not want to change the action list.

elseList

The set of actions performed if the condition defined by the query parameter is not satisfied. You can specify from 0 to 25 actions in this list (limited by AR_MAX_ACTIONS). Specify a list with zero items to define no else actions. Specify NULL for this parameter if you do not want to change the else list.

helpText

The help text associated with the active link. This text can be of any length. Specify NULL for this parameter if you do not want to change the help text.

owner

The owning user for the active link. Specify `NULL` for this parameter if you do not want to change the owner.

changeDiary

The additional change diary text to associate with the active link. This text can be of any length and is appended at the end of any existing text. Existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to add to the change diary.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties are set. See Server object properties.

errorActlinkOptions

Reserved for future use. Set to `NULL`.

errorActlinkName

Reserved for future use. Set to `NULL`.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent, ARDeleteActiveLink, ARGetActiveLink, ARGetListActiveLink, ARGetMultipleActiveLinks. See FreeAR for: `FreeARActiveLinkActionList`, `FreeARInternalIdList`, `FreeARPropList`, `FreeARQualifierStruct`, `FreeARStatusList`.

6.2.123 ARSetApplicationState

Description

Sets the application state (maintenance, test, or production) in the BMC Remedy AR System Application State form.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetApplicationState(
   ARControlStruct *control,
   ARNameType applicationName,
   ARNameType stateName,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

applicationName

The name of the application.

stateName

The value for the state field, **currentStateName**, on the BMC Remedy AR System Application State form. There is one entry per application.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetApplicationState, ARGetListApplicationState.

6.2.124 ARSetCharMenu

Description

Updates the character menu with the indicated name on the specified server. The changes are added to the server immediately and returned to users who request information about character menus. Because character menus operate on clients, individual clients do not receive the updated definition until they reconnect to the form (thus reloading the form from the server).

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Note

The `refreshCode` parameter has no effect on when the updated menu definition is retrieved. Rather, it controls how often the menu contents are retrieved by using the cached menu definition.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetCharMenu(
   ARControlStruct *control,
   ARNameType name,
   ARNameType newName,
   unsigned int *refreshCode,
   ARCharMenuStruct *menuDefn,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

name

The name of the character menu to update.

newName

The new name for the character menu. The names of all character menus on a given server must be unique. Specify `NULL` for this parameter if you do not want to change the name of the character menu.

refreshCode

A value that indicates when the menu is refreshed. This parameter enables you to balance menu consistency with performance.

1:	Refresh only when the form is opened (`AR_MENU_REFRESH_CONNECT`).
2:	Refresh every time the menu is opened (`AR_MENU_REFRESH_OPEN`).
3:	Refresh the first time the menu is opened and every 15 minutes thereafter (`AR_MENU_REFRESH_INTERVAL`).

Specify `NULL` for this parameter if you do not want to change the refresh code.

menuDefn

The definition of the character menu. Specify `NULL` for this parameter if you do not want to change the menu definition.

helpText

The help text associated with the character menu. This text can be of any length. Specify `NULL` for this parameter if you do not want to change the help text.

owner

The owning user for the character menu. Specify `NULL` for this parameter if you do not want to change the owner.

changeDiary

The additional change diary text to associate with the character menu. This text can be of any length and is appended at the end of any existing text. Existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to add to the change diary.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties are set. See Server object properties.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateCharMenu, ARDeleteCharMenu, ARExpandCharMenu, ARGetCharMenu, ARGetListCharMenu. See FreeAR for: `FreeARCharMenuStruct`, `FreeARPropList`, `FreeARStatusList`.

6.2.125 ARSetContainer

Description

Updates the definition for the container with the indicated name on the specified server.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetContainer(
   ARControlStruct *control,
   ARNameType name,
   ARNameType newName,
   ARPermissionList *groupList,
   ARInternalIdList *admingrpList,
   ARContainerOwnerObjList *ownerObjList,
   char *label,
   char *description,
   unsigned int *type,
   ARReferenceList *references,
   ARBoolean removeFlag,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

name

The name of the container to update.

newName

The new name for the container. The names of all containers on a given server must be unique. The system automatically updates all workflow references to the container if you rename it. Specify `NULL` for this parameter if you do not want to change the name of the container.

groupList

A list of zero or more groups who can access this container. Specify an empty group list to define a container accessible only by users with administrator capability. Specify group ID **0** (Public) to provide access to all users. The permission value that you assign for each group determines whether users in that group see the container in the container list.

1:	Users see the container in the container list (`AR_PERMISSIONS_VISIBLE`).
2:	Users do not see the container in the container list (`AR_PERMISSIONS_HIDDEN`).

The group list that you specify replaces all existing group permissions. Specify `NULL` for this parameter if you do not want to change the group list.

admingrpList

A list of zero or more groups who can administer this container (and the referenced objects). If `ownerObj` is not `NULL`, this parameter is ignored and the Subadministrator group list of the owning form is used instead. Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify an empty administrator group list to define a container that can be administered only by users with administrator capability. Specify group ID `0` (Public) to provide administrator capability to all members of the Subadministrator group. The group list that you specify replaces all existing group permissions. Specify `NULL` for this parameter if you do not want to change the administrator group list.

ownerObjList

A list of schemas that own this container. Specify `ARCONOWNER_NONE` for this parameter if you want the container to exist globally. Specify `NULL` for this parameter if you do not want to change the container's owning object. If the object is locked, you can add owners but you cannot delete them.

label

The label for this container. It can be as many as 255 characters long. Specify `NULL` for this parameter if you do not want to change the label.

description

The description for this container. It can be as many as 2000 characters long. Specify `NULL` for this parameter if you do not want to change the description.

type

The type for this container, either `ARCON_GUIDE` or `ARCON_APP`. Specify `NULL` for this parameter if you do not want to change the type.

references

Pointers to the objects (for example, forms or filters) referenced by this container. Specify `NULL` for this parameter if you do not want to change the references.

removeFlag

A flag that specifies how invalid object references are removed. If `FALSE`, references to nonexistent BMC Remedy AR System objects are removed with no error generated. If `TRUE`, an error is generated. Specify `NULL` for this parameter if you do not want to change the flag.

helpText

The help text associated with the container. This text can be of any length. Specify `NULL` for this parameter if you do not want to change the help text.

owner

The owning user for the container. Specify `NULL` for this parameter if you do not want to change the owner.

changeDiary

The additional change diary text to associate with the container. This text can be of any length and is appended at the end of any existing text. Existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to add to the change diary.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties are set. See Server object properties.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateContainer, ARDeleteContainer, ARGetContainer, ARGetListContainer, ARSetSchema. See FreeAR for: `FreeARContainerInfoList`, `FreeARInternalIdList`, `FreeARPermissionList`, `FreeARPropList`, `FreeARReferenceList`, `FreeARStatusList`.

6.2.126 ARSetEntry

Description

Updates the form entry with the indicated ID on the specified server.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

The system updates data based on the access privileges of the user that you specify for the `control` parameter. User permissions are verified for each specified field. The system generates an error if the user does not have write permission for a field or a field does not exist.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetEntry(
   ARControlStruct *control,
   ARNameType schema,
   AREntryIdList *entryId,
   ARFieldValueList *fieldList,
   ARTimestamp getTime,
   unsigned int option,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form containing the entry to update.

entryId

The ID of the entry to update.

Note

The system identifies entries in join forms by concatenating the entry IDs from the member forms. As a result, an entry ID can consist of one or more values of type `AREntryIdType` and, therefore, is represented by the `AREntryIdList` structure.

fieldList

A list of one or more field/value pairs (specified in any order) that identifies the new data for the entry. Values must be of the data type defined for the field or have a data type of `0` (`AR_DATA_TYPE_NULL`). `NULL` values can be specified only for optional fields. An error is generated if a field does not exist or the user specified by the `control` parameter does not have write permission for a field.

getTime

A time stamp that specifies when the entry was last retrieved. The system compares this value with the value in the `Modified Date` core field to determine whether the entry has been changed since the last retrieval. The system updates the entry if the value that you specify is greater than `Modified Date`. If not, the system returns an error. You can either retrieve the entry again and determine whether to apply your changes or specify `0` for this parameter to bypass the time stamp comparison.

option

A value that indicates whether users can update fields specified in the join qualification (only applicable to join forms).

`0`:	Update fields used in the join criteria (thereby causing the entry to no longer appear in the join form) (`AR_JOIN_SETOPTION_NONE`).
`1`:	Prevent update of fields used in the join criteria (`AR_JOIN_SETOPTION_REF`).

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEntry, ARDeleteEntry, ARGetEntry, ARGetListEntry, ARMergeEntry. See FreeAR for: `FreeAREntryIdList`, `FreeARFieldValueList`, `FreeARStatusList`.

6.2.127 ARSetEscalation

Description

Updates the escalation with the indicated name on the specified server. The changes are added to the server immediately and returned to users who request information about escalations.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetEscalation(
   ARControlStruct *control,
   ARNameType name,
   ARNameType newName,
   AREscalationTmStruct *escalationTm,
   ARWorkflowConnectStruct *workflowConnect,
   unsigned int *enable,
   ARQualifierStruct *query,
   ARFilterActionList *actionList,
   ARFilterActionList *elseList,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

name

The name of the escalation to update.

newName

The new name for the escalation. The names of all escalations on a given server must be unique. Specify `NULL` for this parameter if you do not want to change the name of the escalation.

escalationTm

The time specification for evaluating the escalation condition. This parameter can take one of two forms: a time interval that defines how frequently the server checks the escalation condition (in seconds) or a bitmask that defines a particular day (by month or week) and time (hour and minute) for the server to check the condition. Specify `NULL` for this parameter if you do not want to change the escalation time.

workflowConnect

The list of form names the escalation is linked to. The escalation must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to specify the form list. If the object is locked, you can append a form name to the list but you cannot remove a form name from the list.

enable

A flag to enable or disable this escalation. A value of `0` disables the escalation, causing its condition checks and associated actions not to be performed. A value of `1` enables the escalation, causing its conditions to be checked at the specified time interval. Specify `NULL` for this parameter if you do not want to change this flag.

query

A query operation performed when the escalation is executed that determines the set of entries to which the escalation actions (defined by the `actionList` parameter) are applied. Assign an operation value of `0` (`AR_COND_OP_NONE`) to match all form entries. Specify `NULL` for this parameter if you do not want to change the query.

actionList

The set of actions performed for each entry that matches the criteria defined by the `query` parameter. You can specify from 1 to 25 actions in this list (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to change the action list.

elseList

The set of actions performed if no entries match the criteria defined by the `query` parameter. These actions are *not* performed for all non-matching entries. You can specify from 0 to 25 actions in this list (limited by `AR_MAX_ACTIONS`). Specify a list with zero items to define no else actions. Specify `NULL` for this parameter if you do not want to change the else list.

helpText

The help text associated with the escalation. This text can be of any length. Specify `NULL` for this parameter if you do not want to change the help text.

owner

The owning user for the escalation. Specify `NULL` for this parameter if you do not want to change the owner.

changeDiary

The additional change diary text to associate with the escalation. This text can be of any length and is appended at the end of any existing text. Existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to add to the change diary.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties are set. See Server object properties.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`

- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateEscalation, ARDeleteEscalation, ARGetEscalation, ARGetListEscalation. See FreeAR for: `FreeARFilterActionList`, `FreeARPropList`, `FreeARQualifierStruct`, `FreeARStatusList`.

6.2.128 ARSetField

Description

Updates the definition for the form field with the indicated ID on the specified server.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetField(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalId fieldId,
   ARNameType fieldName,
   ARFieldMappingStruct *fieldMap,
   unsigned int *option,
   unsigned int *createMode,
   unsigned int *fieldOption,
   ARValueStruct *defaultVal,
   ARPermissionList *permissions,
   ARFieldLimitStruct *limit,
   ARDisplayInstanceList *dInstanceList,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   unsigned int setFieldOptions,
   ARPropList *objPropList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

schema

The name of the form containing the field to update.

fieldId

The internal ID of the field to update.

fieldName

The new name for the field. The names of all fields and VUIs associated with a given form must be unique. Specify `NULL` for this parameter if you do not want to change the name of the field.

fieldMap

A pointer to a mapping structure that specifies the underlying form that contains the field to be updated. To update a field in a base form, specify a field type of `1` (`AR_FIELD_REGULAR`) in the structure. Otherwise, specify a field type of 2 (`AR_FIELD_JOIN`) and identify the member form (primary or secondary) and field ID for the new field. If the member form is also a join form, map the field to an underlying form. See Mapping fields in schemas. Specify `NULL` for this parameter if you do not want to change the field mapping.

option

A flag that indicates whether users must enter a value in the field.

1:	Required (data fields only) (`AR_FIELD_OPTION_REQUIRED`).
2:	Optional (data fields only) (`AR_FIELD_OPTION_OPTIONAL`).
3:	System (core data fields only) (`AR_FIELD_OPTION_SYSTEM`).
4:	Display-only (nondata fields only) (`AR_FIELD_OPTION_DISPLAY`).

Specify `NULL*` for this parameter if you do not want to change this flag.

createMode

A flag that indicates the permission status for the field when users submit entries. This parameter is ignored for display-only fields.

1:	Any user (including guest users) can enter data in the field when submitting (`AR_FIELD_OPEN_AT_CREATE`).
2:	Only users who have been granted permission can enter data in the field when submitting (`AR_FIELD_PROTECTED_AT_CREATE`).

Specify `NULL` for this parameter if you do not want to change this flag.

fieldOption

A bitmask that indicates whether the field is to be audited or copied when other fields are audited.

Bit 0:	Audit this field. (`AR_FIELD_BITOPTION_AUDIT`)
Bit 1:	Copy this field when other fields in the form are audited. (`AR_FIELD_BITOPTION_COPY`)
Bit 2:	Indicates this field is for Log Key 1. (`AR_FIELD_BITOPTION_LOG_KEY1`)
Bit 3:	Indicates this field is for Log Key 2. (`AR_FIELD_BITOPTION_LOG_KEY2`)
Bits 2 and 3:	Indicates this field is for Log Key 3. (`AR_FIELD_BITOPTION_LOG_KEY3`)

defaultVal

The value to apply if a user submits an entry with no field value (only applicable to data fields). The default value can be as many as 255 bytes in length (limited by `AR_MAX_DEFAULT_SIZE`) and must be of the same data type as the field. Specify `0` (`AR_DEFAULT_VALUE_NONE`) for trim and control fields or to assign no default. Specify `NULL` for this parameter if you do not want to change the default value.

permissions

A list of zero or more groups who can access this field. Specify an empty permission list to define a field accessible only by users with administrator capability. Specify group ID `0` (Public) to provide access to all users. The permission value that you assign for each group determines whether users in that group can modify the field. See Permissions and structures.

The permission list that you specify replaces all existing group privileges. Specify `NULL` for this parameter if you do not want to change the permissions.

limit

The value limits for the field and other properties specific to the field's type. See Defining field limits for a description of the contained structure that applies to the type of field that you are modifying. The limits and properties that you assign must be of the same data type as the field. Specify `0` (`AR_FIELD_LIMIT_NONE`) for trim and control fields or if you do not want to assign any limits or field type-specific properties. Specify `NULL` for this parameter if you do not want to change the field limits and properties.

dInstanceList

A list of zero or more display properties to associate with the field. See ARCreateField for a description of the possible values. You can define both properties common to all form views (VUIs) and display properties specific to particular views. The system includes the field in each view that you specify, regardless of whether you define any display properties for those views. If you do not specify a property for a particular view, the system uses the default value (if defined). Specify a list with zero items to remove *all* display properties for the field. Specify `NULL` for this parameter if you do not want to change this list.

helpText

The help text associated with the field. This text can be of any length. Specify `NULL` for this parameter if you do not want to change the help text.

owner

The owning user for the field. Specify `NULL` for this parameter if you do not want to change the owner.

changeDiary

The additional change diary text to associate with the field. This text can be of any length and is appended at the end of any existing text. Existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to add to the change diary.

setFieldOptions

A bitmask for requesting special treatment of display instance items. You can set either (but not both) of the following bits:

Bit 0:	The server replaces its display instance items for this field that are present in the `ARDisplayInstanceList` argument to ARSetField( ). The server does not modify or replace display instance items for this field that are not present in the list. (`AR_SETFIELD_OPT_PRESERVE_UNLISTED_DISPLAY_INSTANCES`)
Bit 1:	The server deletes the display instance items for this field present in the `ARDisplayInstanceList` arguments to ARSetField( ). The server does not modify, replace, or delete any other display instance item. (`AR_SETFIELD_OPT_DELETE_LISTED_DISPLAY_INSTANCES`)

If both bits are set to 0, the server replaces all display instance items for this field with the values passed in `dInstanceList`.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no object properties are set. See Server object properties and structures.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateField, ARDeleteField, ARGetField, ARGetListField, ARGetMultipleFields, ARSetMultipleFields.
See FreeAR for: `FreeARDisplayInstanceList`, `FreeARFieldLimitList`, `FreeARPermissionList`, `FreeARStatusList`, `FreeARValueStruct`.

6.2.129 ARSetFilter

Description

Updates the filter with the indicated name on the specified server. The changes are added to the server immediately and returned to users who request information about filters.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetFilter(
   ARControlStruct *control,
   ARNameType name,
   ARNameType newName,
   unsigned int *order,
   ARWorkflowConnectStruct *workflowConnect,
   unsigned int *opSet,
   unsigned int *enable,
   ARQualifierStruct *query,
   ARFilterActionList *actionList,
   ARFilterActionList *elseList,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   unsigned int *errorFilterOptions,
   ARNameType errorFilterName,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

name

The name of the filter to update.

newName

The new name for the filter. The names of all filters on a given server must be unique. Specify `NULL` for this parameter if you do not want to change the name of the filter.

order

A value between `0` and `1000` (inclusive) that determines the filter execution order. When multiple filters are associated with a form, the value associated with each filter determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to change the order.

workflowConnect

The list of form names the filter is linked to. The filter must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to specify the form list. If the object is locked, you can append a form name to the list but you cannot remove a form name from the list.

opSet

A bitmask that indicates the form operations that trigger the filter.

Bit 0:	Execute the filter on get operations (`AR_OPERATION_GET`).
Bit 1:	Execute the filter on set operations (`AR_OPERATION_SET`).
Bit 2:	Execute the filter on create operations (`AR_OPERATION_CREATE`).
Bit 3:	Execute the filter on delete operations (`AR_OPERATION_DELETE`).
Bit 4:	Execute the filter on merge operations (`AR_OPERATION_MERGE`).

Specify `NULL` for this parameter if you do not want to change the operation set.

enable

A flag to enable or disable this filter. A value of `0` disables the filter, causing its condition checks and associated actions not to be performed. A value of `1` enables the filter, causing its conditions to be checked for each form operation specified by the `opSet` parameter. Specify `NULL` for this parameter if you do not want to change this flag.

query

A qualification that determines whether the filter is executed. Assign an operation value of `0` (`AR_COND_OP_NONE`) to execute the filter unconditionally. Specify `NULL` for this parameter if you do not want to change the query.

actionList

The set of actions performed if the condition defined by the `query` parameter is satisfied. You can specify from 1 to 25 actions in this list (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to change the action list.

elseList

The set of actions performed if the condition defined by the `query` parameter is not satisfied. You can specify from 0 to 25 actions in this list (limited by `AR_MAX_ACTIONS`). Specify a list with zero items to define no else actions. Specify `NULL` for this parameter if you do not want to change the else list.

helpText

The help text associated with the filter. This text can be of any length. Specify `NULL` for this parameter if you do not want to change the help text.

owner

The owning user for the filter. Specify `NULL` for this parameter if you do not want to change the owner.

changeDiary

The additional change diary text to associate with the filter. This text can be of any length and is appended at the end of any existing text. Existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to add to the change diary.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties are set. See Server object properties.

errorFilterOptions

The error handler options for the filter. Set to `AR_FILTER_ERRHANDLER_ENABLE` to enable an error handler filter. Set to zero to disable. Specify `NULL` for this parameter if you do not want to change error filter options.

errorFilterName

The name of error handler filter for this filter. Specify an empty string for this parameter to remove an existing name. Specify `NULL` if you do not want to change the error filter name.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateFilter, ARDeleteFilter, ARGetFilter, ARGetListFilter. See FreeAR for: `FreeARFilterActionList` ,`FreeARPropList`, `FreeARQualifierStruct`, `FreeARStatusList`.

6.2.130 ARSetGetEntry

Description

Bundles the following API calls into one call:

- `ARSetEntry` — Updates the form entry with the indicated ID on the specified server.
- `ARGetEntry` — Retrieves the form entry with the indicated ID from the specified server. You can retrieve data for specific fields, all (accessible) fields, or no fields (which is useful to verify whether a form has any entries). This function returns only the name, size, and compressed size of attachment fields. Use `ARGetEntryBLOB` to retrieve the contents of the attachment.

After issuing an `ARSetEntry` call to enter a value into a field, BMC Remedy AR System clients always issue an `ARGetEntry` call to ensure that they have the current value for the field (server-side workflow might have modified the value after it was set).

These multiple calls can degrade a client's performance, especially on Wide Area Networks (WANs). To improve performance, use `ARSetGetEntry`, which requires only one call from the client to execute the combined functionality of `ARSetEntry` and `ARGetEntry`.

Because `ARSetGetEntry` includes a query, it cannot be used in bulk API calls.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

The system updates and returns data based on the access privileges of the user that you specify for the `control` parameter. User permissions are verified for each specified field.

The system generates an error if the user does not have write permission for a field or a field does not exist.

The system returns values for accessible fields and warning messages for fields the user cannot access.

Backward compatibility

When an AR System 7.6.02 or later client issues this call to an AR System 7.5.00 or earlier server, the client replaces the `ARSetGetEntry` call with two separate calls: `ARSetEntry` and `ARGetEntry`. In this situation, the client's performance is not improved, but the correct `SetEntry` operation takes place, followed by the correct `GetEntry` operation.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetGetEntry(
   ARControlStruct *control,
   ARNameType schema,
   AREntryIdList *entryId,
   ARFieldValueList *fieldList,
   ARTimestamp getTime,
   unsigned int option,
   ARInternalIdList *idList,
   ARFieldValueList *getFieldList,
   ARStatusList *seStatus
   ARStatusList *geStatus
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

schema

The name of the form containing the entry to update and retrieve.

entryId

The ID of the entry to update and retrieve.

Note

The system identifies entries in join forms by concatenating the entry IDs from the member forms. As a result, an entry ID can consist of one or more values of type `AREntryIdType` and, therefore, is represented by the `AREntryIdList` structure.

fieldList

For `ARSetEntry` functionality, a list of one or more field/value pairs (specified in any order) that identifies the new data for the entry. Values must be of the data type defined for the field or have a data type of `0` (`AR_DATA_TYPE_NULL`). `NULL` values can be specified only for optional fields. An error is generated if a field does not exist or the user specified by the `control` parameter does not have write permission for a field.

getTime

For `ARSetEntry` functionality, a time stamp that specifies when the entry was last retrieved. The system compares this value with the value in the **Modified Date** core field to determine whether the entry has been changed since the last retrieval. The system updates the entry if the value that you specify is greater than **Modified Date**. If not, the system returns an error. You can either retrieve the entry again and determine whether to apply your changes or specify **0** for this parameter to bypass the time stamp comparison.

option

For `ARSetEntry` functionality, a value that indicates whether users can update fields specified in the join qualification (only applicable to join forms).

`0`:	Update fields used in the join criteria (thereby causing the entry to no longer appear in the join form) (`AR_JOIN_SETOPTION_NONE`).
`1`:	Prevent update of fields used in the join criteria (`AR_JOIN_SETOPTION_REF`).

idList

For `ARGetEntry` functionality, a list of zero or more internal IDs that specify the fields to retrieve. Specify `NULL` for this parameter (or zero fields) to retrieve all (accessible) fields. Specify `NULL` for both this parameter and the `fieldList` parameter if you do not want to retrieve any fields. To minimize network traffic, specify only the fields that you need if you do not require the data for all fields. If an attachment field is specified in the list, only its name, size, and compressed size are returned. Use `ARGetEntryBLOB` to retrieve the contents of the attachment.

Return values

fieldList

For `ARGetEntry` functionality, a list of zero or more field/value pairs that identifies the data for the specified entry. The fields are returned in the order specified by `idList`. If the user does not have permission for a specified field or the field does not exist, the system does not return a value for the field/value pair. Specify `NULL` for this parameter if you do not want to retrieve any field data.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

This function returns two statuses:

- `seStatus` — Status list from the `ARSetEntry` operation.

- `geStatus` — Status list from the `ARGetEntry` operation.

If `ARSetEntry` fails, the server does not perform the following `ARGetEntry` call.

See also

ARGetEntry, ARSetEntry.

6.2.131 ARSetImage

Description

Updates the image with the indicated name on the specified server. After the image is updated, the server updates all references and object property timestamps for schemas affected by the change.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetImage(
   ARControlStruct *control,
   ARNameType name,
   ARNameType newName,
   ARImageDataStruct *imageBuf,
   char *imageType,
   char *description,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

name

The name of the image to updated.

newName

The new name for the image. The names of all images on a given server must be unique. Specify `NULL` to leave the name unchanged.

imageBuf

New data for the new image. See Images and structures. Specify `NULL` to leave the image data unchanged.

imageType

The new image encoding type for the image. Valid values are: **BMP**, **GIF**, **JPEG** or **JPG**, and **PNG**. Specify `NULL` to leave the type unchanged.

description

The new description for this image. Specify `NULL` to leave the description unchanged.

helpText

The new help text to be associated with the image. Specify `NULL` to leave the help text unchanged.

owner

The new owner for the image. Specify `NULL` to leave the owner unchanged.

changeDiary

The additional change diary text to associate with the image. This text can be of any length and is appended at the end of any existing text. Existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to add to the change diary.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no new properties are set. See Server object properties and structures.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateImage, ARDeleteImage, ARGetImage, ARGetListImage ARGetMultipleImages. See FreeAR for: `FreeARImageDataStruct`, `FreeARStatusList`.

6.2.132 ARSetImpersonatedUser

Description

Enables plug-ins, the mid tier, and other programs (such as the Email Engine) to run as an administrator but to perform operations as a specific user (with the user's permissions and licensing in effect). For more information, see Impersonating a user.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetImpersonatedUser(
   ARControlStruct *control,
   ARAccessNameType impersonatedUser,
   ARStatusList *status)
```

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

impersonatedUser

The name of the user that the API is impersonating. It is `NULL` if you want to stop impersonating.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.133 ARSetLogging

Description

Activates and deactivates client-side logging of server activity.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

Users with permission to perform client-side logging. (See Client-side ARAPILOGGING.)

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARSetLogging(
   ARControlStruct *control,
   ARULong32 logTypeMask,
   ARULong32 whereToWriteMask,
   FILE *filePtr,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

logTypeMask

A bitmask that indicates the type of logging to set. This is for SQL, filter, API, and plug-in server logging. The possible values are:

Bit 1:	Database or SQL logging (AR_DEBUG_SERVER_SQL).
Bit 2:	Filter logging (AR_DEBUG_SERVER_FILTER) .
Bit 5:	API logging (AR_DEBUG_SERVER_API).
Bit 18:	Plug-in logging (AR_DEBUG_SERVER_PLUGIN).

whereToWriteMask

A value that indicates where to return log information. The possible values are

1:	Logs to a file (AR_WRITE_TO_FILE).
2:	Logs as part of an ARStatusList (AR_WRITE_TO_STATUS_LIST).

filePtr

A file handle to the log file. It is assumed the file has already been opened. A `NULL` value indicates that no writing be done to the file.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

If the `whereToWriteMask` input parameter is set to `AR_WRITE_TO_STATUS_LIST`, the message number is of type `messageNum` (number of the status structure), and everything that is logged is of type `AR_NOTE_LOG_INFO`.

6.2.134 ARSetMultipleFields

Description

Updates the definition for a list of fields with the specified IDs on the specified form on the specified server. A call to this function produces the same result as a sequence of `ARSetField` calls to update the individual fields, but it can be more efficient because it requires only one call from the client to the BMC Remedy AR System server and because the server can perform multiple database operations in a single transaction and avoid repeating operations such as those performed at the end of each individual call.

The list input arguments define the field updates the server performs. Each list type consists of a count and a pointer to an array of values. If all values in a list are `NULL`, the argument can be `NULL` or a zero-length list. All list counts that are not zero must be equal.

If a failure occurs when modifying an individual field definition, the error is recorded in the `setFieldStatusList` return value, `AR_WARN_SET_FIELD_FAILED` is added to the `status` return value (if it is not already recorded), and the server continues to modify the other fields in the list. If errors occur only during individual field updates, no failure is recorded in the `status` return value, so the call is considered successful.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
```

```
#include "arextern.h"
#include "arstruct.h"

int ARSetMultipleFields(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalIdList *fieldIdList,
   ARNamePtrList *fieldNameList,
   ARFieldMappingPtrList *fieldMapList,
   ARUnsignedIntList *optionList,
   ARUnsignedIntList *createModeList,
   ARUnsignedIntList *fieldOptionList,
   ARValuePtrList *defaultValList,
   ARPermissionListPtrList *permissionListList,
   ARFieldLimitPtrList *limitList,
   ARDisplayInstanceListPtrList *dInstanceListList,
   ARTextStringList *helpTextList,
   ARAccessNamePtrList *ownerList,
   ARTextStringList *changeDiaryList,
   ARUnsignedIntList *setFieldOptionList,
   ARPropListList *objPropListList,
   ARStatusListList *setFieldStatusList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

schema

The name of an existing form on the server. The fields must all be on that form.

fieldIdList

A list of the internal IDs of the fields to update.

fieldNameList

A list of pointers to new names for the fields. The names of all fields and VUIs associated with a given form must be unique.

To leave the name of a field unchanged, put a `NULL` field name pointer in corresponding list item. To leave all field names unchanged, pass `NULL` for this parameter.

fieldMapList

A list of pointers to mapping structures that specify the underlying form for each field to be updated. To update a fields in a base form, specify a field type of 1 (`AR_FIELD_REGULAR`) in all structures in the list. Otherwise, specify a field type of 2 (`AR_FIELD_JOIN`) and identify the member form (primary or secondary) and field ID for the new field. If the member form is also a join form, create fields in all nested join forms until you can map the field to an underlying base form. See Mapping fields in schemas.

To leave the mapping for a field unchanged, put a `NULL` field mapping pointer in corresponding list item. To leave all field mappings unchanged, pass `NULL` for this parameter.

optionList

A list of pointers to values that specify whether users must enter a value in the field.

1:	Required (data fields only) (`AR_FIELD_OPTION_REQUIRED`).
2:	Optional (data fields only) (`AR_FIELD_OPTION_OPTIONAL`).
4:	Display-only (data fields only). Works like an optional field but doesn't write to the database (`AR_FIELD_OPTION_DISPLAY`).

To leave the option unchanged for a field, put a `NULL` pointer in the corresponding list item. To leave all options unchanged, pass `NULL` for this parameter.

createModeList

A list of pointers to values that specify the permission status for the field when users submit entries. The list entry is ignored for display-only fields.

1:	Any user (including guest users) can enter data in the field when submitting (`AR_FIELD_OPEN_AT_CREATE`).
2:	Only users who have been granted permission can enter data in the field when submitting (`AR_FIELD_PROTECTED_AT_CREATE`).

To leave the create mode unchanged for a field, put a `NULL` pointer in the corresponding list item. To leave all field create modes unchanged, pass `NULL` for this parameter.

fieldOptionList

A list of pointers to bitmasks that specify whether the field is to be audited or copied when other fields are audited.

Bit 0:	Audit this field. (`AR_FIELD_BITOPTION_AUDIT`)
Bit 1:	Copy this field when other fields in the form are audited. (`AR_FIELD_BITOPTION_COPY`)
Bit 2:	Indicates this field is for Log Key 1. (`AR_FIELD_BITOPTION_LOG_KEY1`)

Bit 3:	Indicates this field is for Log Key 2. (`AR_FIELD_BITOPTION_LOG_KEY2`)
Bits 2 and 3:	Indicates this field is for Log Key 3. (`AR_FIELD_BITOPTION_LOG_KEY3`)

To leave the field option unchanged for a field, put a `NULL` pointer in the corresponding list item. To leave all field options unchanged, pass `NULL` for this parameter.

defaultValList

A list of pointers to the values to apply to the corresponding data field when a user submits an entry with no field value. The default value can be as many as 255 bytes in length (limited by `AR_MAX_DEFAULT_SIZE`) and must be of the same data type as the field. To remove the default value for a field, put `AR_DEFAULT_VALUE_NONE` in the corresponding list item.

For trim, control, panel, and panel holder fields or to leave the default value for a data field unchanged, put `NULL` in the corresponding list item. To leave all default values unchanged, pass `NULL` for this parameter.

permissionsListList

A list of pointers to lists of zero or more groups who can access the corresponding field. To leave the permission list unchanged for a field, put a `NULL` pointer in the corresponding list item. To make a field accessible only by users with administrator capability, put an empty permission list in the corresponding list item. To provide access to all users, specify group ID `0` (Public). The permission value that you assign for each group determines whether users in that group can modify the field. See Permissions and structures.

To leave all permission lists unchanged, pass `NULL` for this parameter.

limitList

A list of pointers to the value limits for the corresponding field and other properties specific to the field's type. See Defining field limits for a description of the contained structure that applies to the type of field that you are creating. The limits and properties must be of the same data type as the field. For trim and control fields and to clear all limits and properties for field, put `AR_FIELD_LIMIT_NONE` in the corresponding list item.

To leave the limits and properties for a field unchanged, put `NULL` in the corresponding list item. To leave all field limits unchanged, pass `NULL` for this parameter.

dInstanceListList

A list of pointers to lists of zero or more display properties to associate with the corresponding field. You can define both display properties common to all form views (VUIs) and display properties specific to particular views. The system includes the field in each view that you specify, regardless of whether you define any display properties for those views. If you do not specify a property for a particular view, the system uses the default value (if defined). To remove all display instance items for a field, put a pointer to a list of zero items in the corresponding list item. To leave the display instance items for a field unchanged, put `NULL` in the corresponding list item.

To leave all field display instance items unchanged, pass `NULL` for this parameter.

Compare the description of `setFieldOptionList` to understand how that parameter can affect the behavior of `dInstanceListList`.

helpTextList

A list of pointers to the help text associated with the corresponding field. The text can be of any length. To leave the help text for a field unchanged, put `NULL` in the corresponding list item.

To leave all field help text unchanged, pass `NULL` for this parameter.

ownerList

A list of pointers to the owner for the corresponding field. To leave the owner for a field unchanged, put `NULL` in the corresponding list item.

To leave all field owners unchanged, pass `NULL` for this parameter.

changeDiaryList

A list of pointers to the additional change diary text to add to the change diary of the corresponding field. The text can be of any length and is appended at the end of any existing change diary text. The existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. To leave the change diary for a field unchanged, put `NULL` in the corresponding list item.

To leave all field change diaries unchanged, pass `NULL` for this parameter.

setFieldOptionList

A list of bitmasks for requesting special treatment of display instance items. You can set either (but not both) of the following bits:

Bit 0:	The server replaces its display instance items for this field that are present in the `ARDisplayInstanceList` argument to ARSetField( ). The server does not modify or replace display instance items for this field that are not present in the list. (`AR_SETFIELD_OPT_PRESERVE_UNLISTED_DISPLAY_INSTANCES`)
Bit 1:	The server deletes the display instance items for this field present in the `ARDisplayInstanceList` arguments to ARSetField( ). The server does not modify, replace, or delete any other display instance item. (`AR_SETFIELD_OPT_DELETE_LISTED_DISPLAY_INSTANCES`)

If both bits are set to 0, the server replaces all display instance items for this field with the values passed in `dInstanceListList`.

To specify default processing for all fields, pass `NULL` for this parameter.

objPropListList

A list of server object property lists that correspond to the fields in `fieldIdList`. If this parameter is set to `NULL`, no properties are set. See Server object properties and structures.

Return values

setFieldStatusList

A list of status list structures corresponding to each field being modified. If an error occurs during the modification of a field, the corresponding status reports it. If any error occurs during any field modification, the `status` list contains a warning, `AR_WARN_SET_FIELD_FAILED`, to indicate that the detailed error information is in this parameter.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. The warning, `AR_WARN_SET_FIELD_FAILED`, indicates that an error occurred during one or more of the field modifications. See the **setFieldStatusList** parameter for detailed error information. For a description of all possible values, see Error checking.

See also

ARCreateMultipleFields, ARSetField. See FreeAR for: `FreeARAccessNamePtrList`, `FreeARDisplayInstanceListPtrList`, `FreeARFieldLimitPtrList`, `FreeARFieldMappingPtrList`, `FreeARInternalIdList`, `FreeARNamePtrList`, `FreeARPermissionListPtrList`, `FreeARStatusList`, `FreeARTextStringList`, `FreeARUnsignedIntList`, `FreeARValuePtrList`.

6.2.135 ARSetSchema

Description

Updates the definition for the form with the indicated name on the specified server. If the schema is locked, only the indexList and the defaultVui can be set.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetSchema(
   ARControlStruct *control,
   ARNameType name,
   ARNameType newName,
   ARCompoundSchema *schema,
   ARSchemaInterheritanceList *schemaInheritanceList,
   ARPermissionList *groupList,
   ARInternalIdList *admingrpList,
   AREntryListFieldList *getListFields,
   ARSortList *sortList,
   ARIndexList *indexList,
   ARArchiveInfoStruct *archiveInfo,
   ARAuditInfoStruct *auditInfo,
   ARNameType defaultVui,
   char *helpText,
   ARAccessNameType owner,
   char *changeDiary,
   ARPropList *objPropList,
   unsigned int setOption,
   char *objectModificationLogLabel,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

name

The name of the form to update.

newName

The new name for the form. The names of all forms on a given server must be unique. The system automatically updates all workflow references to the form if you rename it. Specify **NULL** for this parameter if you do not want to change the name of the form.

schema

The form type (base form or join form). See ARCreateSchema for a description of the possible values. You cannot change a join form to a base form or vice versa. You can, however, modify the join criteria for a join form. Specify `NULL` for this parameter if you do not want to change the form information.

schemaInheritanceList

This is reserved for future use. Specify `NULL`.

groupList

A list of zero or more groups who can access this form. Specify an empty group list to define a form accessible only by users with administrator capability. Specify group ID `0` (Public) to provide access to all users. The permission value that you assign for each group determines whether users in that group see the form in the form list.

`1`:	Users see the form in the form list (`AR_PERMISSIONS_VISIBLE`).
`2`:	Users do not see the form in the form list (`AR_PERMISSIONS_HIDDEN`).

The group list that you specify replaces all existing group permissions. Specify `NULL` for this parameter if you do not want to change the group list.

admingrpList

A list of zero or more groups who can administer this form (and the associated filters, escalations, and active links). Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify an empty administrator group list to define a form that can be administered only by users with administrator capability. Specify group ID `0` (Public) to provide administrator capability to all members of the Subadministrator group. The group list that you specify replaces all existing group permissions. Specify `NULL` for this parameter if you do not want to change the administrator group list.

getListFields

A list of zero or more fields that identifies the default query list data for retrieving form entries. The combined length of all specified fields, including separator characters, can be as many as 128 bytes (limited by `AR_MAX_SDESC_SIZE`). Specify zero fields to assign the Short-Description core field as the default query list data. Specifying a `getListFields` argument when calling the `ARGetListEntry` function overrides the default query list data. Specify `NULL` for this parameter if you do not want to change the default query list fields.

sortList

A list of zero or more fields that identifies the default sort order for retrieving form entries. Specifying a `sortList` argument when calling the `ARGetListEntry` function overrides the default sort order. Specify `NULL` for this parameter if you do not want to change the default sort fields.

indexList

The set of zero or more indexes to create for the form. You can specify from 1 to 16 fields for each index (limited by `AR_MAX_INDEX_FIELDS`). Diary fields and character fields larger than 255 bytes cannot be indexed. Specify `NULL` for this parameter if you do not want to change the index list.

archiveInfo

If a form is to be archived, this is the archive information for the form. Specify `NULL` for this parameter if you do not want to create or set the archive information. These are the values for archive type:

`0`:	Deletes the archive settings for the selected form. The selected form is not archived (`AR_ARCHIVE_NONE`).

1:	Entries on the selected form are copied to the archive form (AR_ARCHIVE_FORM).
2:	Entries on the selected form are deleted from the source form (AR_ARCHIVE_DELETE).
4:	Entries on the selected form are copied to an XML format file (AR_ARCHIVE_FILE_XML).
8:	Entries on the selected form are copied to an ARX format file (AR_ARCHIVE_FILE_ARX).
32:	Attachment fields are not copied to the archive form. (AR_ARCHIVE_NO_ATTACHMENTS).
64:	Diary fields are not copied to the archive form. (AR_ARCHIVE_NO_DIARY).

In addition to the archive type, `archiveInfo` also stores the form name (if archive type is AR_ARCHIVE_FORM), time to trigger the archive, and the archive qualification criteria. For an archive form, only the name of the source form is maintained, and none of this archive information can be set.

auditInfo

If a form is to be audited, this is the audit information for the form. Specify NULL for this parameter if you do not want to create or set the audit information. These are the values for audit type:

0:	The selected form is not to be audited. (AR_AUDIT_NONE).
1:	Audit fields and copy fields on the selected form are copied to the audit shadow form (AR_AUDIT_COPY).
2:	Audit fields and copy fields on the selected form are copied to the audit log form (AR_AUDIT_LOG).

In addition to the audit type, `auditInfo` also stores the form name (if audit type is AR_AUDIT_COPY or AR_AUDIT_LOG) and the audit qualification criteria.

defaultVui

The label of the default view. Specify NULL for this parameter if you do not want to set this value.

helpText

The help text associated with the form. This text can be of any length. Specify NULL for this parameter if you do not want to change the help text.

owner

The owning user for the form. Specify NULL for this parameter if you do not want to change the owner.

changeDiary

The additional change diary text associated with the form. This text can be of any length and is appended at the end of any existing text. Existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to add to the change diary.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no properties are set. See Server object properties.

setOption

Reserved for future use.

objectModificationLogLabel

The version control label that the API function must apply to the object. If the label name does not exist on the server, the function creates it.

Ripple actions

Rename and Delete operations typically change multiple objects in addition to their primary target object. The Rename or Delete function must apply the version control label to *all* the objects that it affects.

Multiple API calls for a single user action

Some user actions trigger a sequence of API calls. In that situation, the last API call in the sequence applies the version control label to the object. Thus, clients that create forms (like BMC Remedy Developer Studio does) should provide the label only to the last call. For example, when a form is created, a client might issue these calls:

- `ARCreateSchema`
- `ARCreateMultipleFields`
- `ARSetVUI`
- `ARSetVUI`
- `ARSetSchema`

In this case, the `objectModificationLogLabel` value should be passed only to the last call, `ARSetSchema`, even though the user provides the label during the `ARCreateSchema` operation.

Operations on label-related forms

Version control labels cannot be applied to these forms:

- AR System Version Control: Label
- AR System Version Control: Labeled Object
- AR System Version Control: Object Modification Log

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateField, ARCreateSchema, ARDeleteSchema, ARGetListEntry, ARGetListAlertUser, ARGetSchema. See FreeAR for: `FreeARCompoundSchema, FreeAREntryListFieldList, FreeARIndexList, FreeARInternalIdList, FreeARPermissionList, FreeARPropList, FreeARSortList, FreeARStatusList`.

6.2.136 ARSetServerInfo

Description

Updates the indicated configuration information for the specified server.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"

int ARSetServerInfo(
   ARControlStruct *control,
   ARServerInfoList *serverInfo,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

serverInfo

The information to update on the server (see the following Server options section). The system returns error messages for server options not updated due to error.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

Server options

Some server options can be set and retrieved. Others can only be retrieved and still others can only be set.

- For a complete list of options that can be set with the `ARSetServerInfo` call and retrieved with the `ARGetServerInfo` call, see the server options descriptions for ARGetServerInfo.
- For a complete list of options that can only be retrieved, see the server option descriptions for ARGetServerInfo.
- You can set the value of the `AR_SERVER_INFO_DB_PASSWORD` option, but you cannot retrieve its value.
  `AR_SERVER_INFO_DB_PASSWORD`:
  The database password associated with the **ARSystem** (applicable to Oracle, Microsoft SQL Server, and Sybase databases). The encrypted password value is stored in the configuration file.

See also

ar.conf file, ARGetServerInfo. See FreeAR for: `FreeARServerInfoList, FreeARStatusList`.

6.2.137 ARSetServerPort

Description

Specifies the port that your program uses to communicate with the BMC Remedy AR System server and whether to use a private queue.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
```

```
int ARSetServerPort(
   ARControlStruct *control,
   ARNameType server,
   int port,
   int rpcProgramNum,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

server

The name of the server to update.

port

The port number that your program uses to communicate with the BMC Remedy AR System server. If you do not specify this parameter or provide `0` for the port number, your program uses the port number supplied by the portmapper. This parameter is overridden by the `ARTCPPORT` environment variable.

rpcProgNum

The **RPC** program number of the server. Specify `390600` to use the admin queue, a number from `390621` to `390634` or `390636` to `390669` or `390680` -`390694` to use a private queue, or `0` (default) to use the fast or list server queue. This parameter is overridden by the `ARRPC` environment variable.

To retrieve server statistics information with the `ARGetServerStatistics` call, you can also set the RPC Program number to `390619`, `390620`, or `390635`.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.138 ARSetSessionConfiguration

Description

Sets an API session variable.
If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

This operation can be performed by all users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetSessionConfiguration(
   ARControlStruct *control,
   unsigned int variableId,
   ARValueStruct *variableValue,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **sessionId** fields are required.

variableId

Identification of the variable type:

1:	Maximum size for a single response (`AR_SESS_CHUNK_RESPONSE_SIZE`).
2:	Timeout for normal operations (`AR_SESS_TIMEOUT_NORMAL`).
3:	Timeout for long operations (`AR_SESS_TIMEOUT_LONG`).
4:	Timeout for extra long operations (`AR_SESS_TIMEOUT_XLONG`).
5:	Socket number to lock to (`AR_SESS_LOCK_TO_SOCKET_NUMBER`).
6:	Flag with Boolean value that indicates if the session is pooled (`AR_SESS_POOLED`).
7:	API program client type (`AR_SESS_CLIENT_TYPE`).
8:	Type of VUI view (`AR_SESS_VUI_TYPE`).
9:	Flag with Boolean value that indicates if the user would like to override the session from the previous IP (`AR_SESS_OVERRIDE_PREV_IP`).
`10`:	The name of the API command log (`AR_SESS_API_CMD_LOG`).
`11`:	The name of the API result log (`AR_SESS_API_RES_LOG`).

12:	Overlay group used at design time (AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP).
13:	Overlay group used at runtime (AR_SESS_CONTROL_PROP_API_OVERLAYGROUP).
14:	Granular overlay mode used at runtime. For extended properties, determines whether full lists or only extensions are retrieved. (AR_SESS_CONTROL_PROP_GRANULAR_OVERLAY_RETRIEVE_MODE).

variableValue

Configuration variable value:

- AR_SESS_CHUNK_RESPONSE_SIZE (integer): An integer value for the maximum data packet size returned from the server to the client in one transmission.
- AR_SESS_TIMEOUT_NORMAL (integer): An integer value for the client timeout value used in fast server operations.
- AR_SESS_TIMEOUT_LONG (integer): An integer value for the client timeout value used in list server operations.
- AR_SESS_TIMEOUT_XLONG (integer): An integer value for the client timeout value used in definition updating and in import/export operations.
- AR_SESS_LOCK_TO_SOCKET_NUMBER (integer): An integer value for the socket number that the client locks to for all server interaction.
- AR_SESS_POOLED (integer): An integer value for the flag that indicates if the session is pooled. Valid values for this option are 0 (No) and 1 (Yes).
- AR_CLIENT_TYPE_* (integer): An integer value for the client type. For more information, see AR_CLIENT_TYPE_* in the **ar.h** file.
  Range of values for user-defined, client-type variables:
  AR_CLIENT_TYPE_END_OF_RESERVED_RANGE: > 5000.
- AR_SESS_VUI_TYPE (integer): An integer value for the type of VUI. For more information, see AR_VUI_TYPE_* in the **ar.h** file.
- AR_SESS_OVERRIDE_PREV_IP (integer): An integer value for the flag that indicates if the session can be overridden. Valid values for this option are 0 (No) and 1 (Yes).
- AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP (character): A character value that indicates whether to use an object's real or resolved name in administrative and design-time operations (ARCreate, ARDelete, and ARSet calls) and whether to perform the operation only on objects in a specified overlay group. (Unmodified objects are members of all groups, so they are included in all operations regardless of the value of this variable.)
  Valid values for this option are the following strings:

• None *or* 0 — Use real names; perform operation on origin (overlaid and unmodified) objects.
• overlayGroupID — Use resolved names; perform operation on objects in the specified overlay group. (In AR System 7.6.04, you can only specify group ID 1.)
• -1 — (Base Development mode) Use real names; perform operation on origin (overlaid and unmodified) objects.
• -2 — (Full Development mode) Use real names; perform operation on all objects (overlay, overlaid, unmodified, and custom).

- AR_SESS_CONTROL_PROP_API_OVERLAYGROUP (character): A character value that indicates whether to use an object's real or resolved name in runtime operations (ARGet, ARGetList, and ARGetMultiple calls) and whether to perform the operation only on custom and overlay objects in a specified overlay group. (Unmodified objects are members of all groups, so they are included in all operations regardless of the value of this variable.)
  Valid values for this option are the following strings:

• None *or* 0 — Use real names; perform operation on origin (overlaid and unmodified) objects.
• overlayGroupID — Use resolved names; perform operation on objects in the specified overlay group. (In AR System 7.6.04, you can only specify group ID **1**.)
• -1 — (Base Development mode) Use real names; perform operation on origin (overlaid and unmodified) objects.
• -2 — (Full Development mode) Use real names; perform operation on all objects (overlay, overlaid, unmodified, and custom).

- AR_SESS_CONTROL_PROP_GRANULAR_OVERLAY_RETRIEVE_MODE (int): An integer value that defines what type of object definition is returned. Valid values for this option are:
  - 0 (the default): All Get<object> calls return the complete object definition.
  - 1: For grains of overlays that can be extended (form lists, permissions, and indexes), Get<object> calls return only the values used to extend or overwrite the values retrieved from the origin object.
    For all other grains, the inherited or overwritten values are returned.
- AR_SESS_API_CMD_LOG (char): A character value for the name of the API client side logging command file.
- AR_SESS_API_RES_LOG (char): A character value for the name of the API client side logging result file.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetSessionConfiguration.

6.2.139 ARSetSupportFile

Description

Sets a support file in the BMC Remedy AR System.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetSupportFile (
   ARControlStruct *control,
   unsigned int fileType,
   ARNameType name,
   ARInternalId id2,
   ARInternalId fileId,
   FILE *filePtr,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

fileType

The numerical value for the type of file, and the type of object the file is related to. Specify `1` (`AR_SUPPORT_FILE_EXTERNAL_REPORT`) for an external report file associated with an active link.

name

The name of the object the file is associated with, usually a form.

id2

The associated identifier if the object identifier is not enough.

fileId

The unique identifier of a file within its object.

filePtr

A pointer to the support file to be set in the system. If you are using Windows, you must open the file in binary mode.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateAlertEvent, ARCreateSupportFile, ARDeleteActiveLink, ARDeleteSupportFile, ARGetActiveLink, ARGetListSupportFile, ARGetSupportFile, ARSetActiveLink.

6.2.140 ARSetVUI

Description

Updates the form view (VUI) with the indicated ID on the specified server.

If you perform a set operation on an overlay and you provide values for an inherited grain, the values are ignored. Only values for extended or overwritten grains of the overlay are set.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSetVUI(
   ARControlStruct *control,
   ARNameType schema,
   ARInternalId vuiId,
   ARNameType vuiName,
   ARLocaleType locale,
   unsigned int *vuiType,
   ARPropList **objPropList*, char *helpText, ARAccessNameType owner, char *changeDiary,
ARPropList *objPropList, ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

If a valid overlay group is specified in the control record, the `ARSet*` function operates on the object that belongs to that group. If no group is specified, the function operates on the origin object. To specify whether to use an object's real or resolved name in an operation and whether to perform the operation only on objects in a specified overlay group, use the `AR_SESS_CONTROL_PROP_DESIGN_OVERLAYGROUP` variable of the `ARSetSessionConfiguration` function (see ARSetSessionConfiguration).

schema

The name of the form containing the VUI to update.

vuiId

The internal ID of the VUI to update.

vuiName

The new name for the VUI. The names of all VUIs and fields associated with a given form must be unique. Specify `NULL` for this parameter if you do not want to change the name of the VUI.

locale

The locale of the VUI. Specify `NULL` for this parameter if you do not want to change the name of the VUI.

vuiType

The type of VUI. Specify `NULL` for this parameter if you do not want to retrieve the VUI type.

`0:`	No VUI type (`AR_VUI_TYPE_NONE`).
`1:`	Windows type (`AR_VUI_TYPE_WINDOWS`)* - fields in BMC Remedy User.
`2:`	Web relative type (`AR_VUI_TYPE_WEB`) - fields in view can be adjusted.
`3:`	Web absolute type (`AR_VUI_TYPE_WEB_ABS_POS`) - fields in view are fixed.

Field marked with an asterisk (*) is available for legacy environments that use BMC Remedy User, which is no longer shipped with BMC Remedy AR System.

dPropList

A list of zero or more display properties to associate with the VUI. See ARCreateVUI for a description of the possible values. Specify `0` (`AR_DPROP_NONE`) to assign no display properties. Specify **NULL** for this parameter if you do not want to change this list.

helpText

The help text associated with the VUI. This text can be of any length. Specify `NULL` for this parameter if you do not want to change the help text.

owner

The owning user for the VUI. Specify `NULL` for this parameter if you do not want to change the owner.

changeDiary

The additional change diary text to associate with the VUI. This text can be of any length and is appended at the end of any existing text. Existing change diary text cannot be deleted or changed. The server adds the user making the change and a time stamp when it saves the change diary. Specify `NULL` for this parameter if you do not want to associate change diary text with this object.

objPropList

A list of server object properties. If this parameter is set to `NULL`, no new properties are set. See Server object properties.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateVUI, ARDeleteVUI, ARGetVUI, ARGetListVUI. See FreeAR for: `FreeARPropList`, `FreeARStatusList`.

6.2.141 ARSignal

Description

Causes the server to reload information.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARSignal(
   ARControlStruct *control,
   ARSignalList *signalList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

signalList

The list of signals. The options for this parameter are:

`AR_SIGNAL_CONFIG_CHANGED`	Causes the server to reload information from its configuration file (**ar.conf** on UNIX, **ar.cfg** on Windows).
`AR_SIGNAL_GROUP_CACHE_CHANGED`	Causes the server to reload group and data dictionary information from the database.
`AR_SIGNAL_LICENSE_CHANGED`	Causes the server to reload license information.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.142 ARTermination

Description

Performs environment-specific cleanup routines and disconnects from the specified BMC Remedy AR System session. All API programs that interact with the BMC Remedy AR System should call this function upon completing work in a given session. Calling this function is especially important in environments that use floating licenses. If you do not disconnect from the server, your license token is unavailable for other users for the defined time-out interval.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
```

```
#include "arstruct.h"

int ARTermination (
   ARControlStruct *control,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARInitialization. See FreeAR for: FreeARStatusList.

6.2.143 ARValidateFormCache

Description

Retrieves key information from a cached definition (such as form name, form change time, active link change time, active link guide change time, and menu change time) and returns information about any changes to the definition given this information. This function also returns information about updates to the current user performing a test. This information causes the cache to reload to insure accurate definitions.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARValidateFormCache(
   ARControlStruct *control,
   ARNameType form,
   ARTimestamp mostRecentActLink,
   ARTimestamp mostRecentMenu,
   ARTimestamp mostRecentGuide,
   ARTimestamp *formLastModified,
   int *numActLinkOnForm,
   int *numActLinkSince,
   ARNameList *menuSinceList,
   ARTimestamp *groupsLastChanged,
   ARTimestamp *userLastChanged,
   ARNameList *guideSinceList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

form

The name of the form to validate changes in.

mostRecentActLink

A time stamp that you provide for the most recent modification time that you know about of an active link for the form.

mostRecentMenu

A user supplied time stamp that provides the most recent modification time for the form menu.

mostRecentGuide

A user supplied time stamp that provides the most recent modification time for an active link guide for the form.

Return values

formLastModified

A time stamp that identifies the last modification time of the form. Specify `NULL` for this parameter if you do not want to retrieve this value.

numActLinkOnForm

An integer that indicates the number of active links associated with the form that are accessible by the current user. Specify `NULL` for this parameter if you do not want to retrieve this value.

numActLinkSince

An integer that indicates the number of active links associated with the form that are accessible to the current user and that have been modified since the time indicated by `mostRecentActLink`. Specify `NULL` for this parameter if you do not want to retrieve this value.

menuSinceList

A list of menus changed since the time specified in `mostRecentMenu`. Specify `NULL` for this parameter if you do not want to retrieve this value.

groupsLastChanged

A time stamp that specifies the last change made to any Group on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

userLastChanged

A time stamp that specifies the last change made to the current user. Specify `NULL` for this parameter if you do not want to retrieve this value.

guideSinceList

A list of active link guides changed since the time specified in the `mostRecentGuide` argument. Specify `NULL` for this parameter if you do not want to retrieve this value.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetListActiveLink, ARGetListCharMenu, ARGetListContainer, ARGetSchema, ARGetServerInfo. See FreeAR for: `FreeARStatusList`.

6.2.144 ARValidateLicense

Description

Confirms whether the current server holds a license of the specified type.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"
```

```
int ARValidateLicense(
   ARControlStruct *control,
   ARLicenseNameType licenseType,
   ARLicenseValidStruct *licenseValid,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

licenseTypeList

A list of the licenses that the function validates, including information such as license key, license type, expiration date, number of licenses, and qualifier. The system does not support a `NULL` value in this option.

Return values

licenseValid

License information, such as license key, license type, number of licenses, expiration date, and qualification.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateLicense, ARDeleteLicense, ARGetListLicense, ARValidateMultipleLicenses. See FreeAR for: `FreeARLicenseValidStruct, FreeARStatusList`.

6.2.145 ARValidateMultipleLicenses

Description

Checks whether the current server holds a license for several specified license types. This function performs the same action as `ARValidateLicense`, but it is easier to use and more efficient than validating licenses one by one.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARValidateMultipleLicenses(
   ARControlStruct *control,
   ARLicenseNameList *licenseTypeList,
   ARLicenseValidList *licenseValidList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

licenseTypeList

A list of the licenses that the function validates, including information such as license key, license type, expiration date, number of licenses, and qualifier. The system does not support a `NULL` value in this option.

Return values

licenseValidList

A list of licenses, including information such as license key, license type, number of licenses, expiration date, and qualification.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARCreateLicense, ARDeleteLicense, ARGetListLicense, ARValidateLicense, ARValidateMultipleLicenses.
See FreeAR for: `FreeARLicenseNameList, FreeARLicenseValidList, FreeARStatusList`.

6.2.146 ARVerifyUser

Description

Checks the cache on the specified server to determine whether the specified user is registered with the current server.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARVerifyUser(
   ARControlStruct *control,
   ARBoolean *adminFlag,
   ARBoolean *subAdminFlag,
   ARBoolean *customFlag,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

adminFlag

A flag that indicates whether the specified user is a member of the Administrator group. The system returns `1` (`TRUE`) if the user is a member of the Administrator group. The system returns `0` (`FALSE`) if the user is not a member of the Administrator group, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

subAdminFlag

A flag that indicates whether the specified user is a member of the Subadministrator group. The system always returns `1` (`TRUE`) if the user is a member of the Subadministrator group. The system returns `0` (`FALSE`) if the user is not a member of the Subadministrator group, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

customFlag

A flag that indicates whether the specified user is a member of the Customize group. The system always returns `1` (`TRUE`) if the user is a member of the Customize group. The system returns `0` (`FALSE`) if the user is not a member of the Customize group, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

See FreeAR for: `FreeARStatusList.`

6.2.147 ARVerifyUser2

Description

Checks the cache on the specified server to determine whether the specified user is registered with the current server.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARVerifyUser2(
   ARControlStruct *control,
   ARBoolean *adminFlag,
   ARBoolean *subAdminFlag,
   ARBoolean *customFlag,
   ARBoolean *stadminFlag,
   ARBoolean *stsubadminFlag,
   ARBoolean *overlayFlag,
   ARBoolean *boverlayFlag,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

adminFlag

A flag that indicates whether the specified user is a member of the Administrator group. The system returns `1 (TRUE)` if the user is a member of the Administrator group. The system returns `0 (FALSE)` if the user is not a member of the Administrator group, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

subAdminFlag

A flag that indicates whether the specified user is a member of the Subadministrator group. The system always returns `1 (TRUE)` if the user is a member of the Subadministrator group. The system returns `0 (FALSE)` if the user is not a member of the Subadministrator group, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

customFlag

A flag that indicates whether the specified user is a member of the Customize group. The system always returns `1 (TRUE)` if the user is a member of the Customize group. The system returns `0 (FALSE)` if the user is not a member of the Customize group, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

structAdminFlag

A flag that indicates if the user is assigned to the Struct Admin group. The system returns `1 (TRUE)` if the user is a member of the Struct Admin group. The system returns `0 (FALSE)` if the user is not a member of the structAdmin group, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

structSubAdminFlag

A flag that indicates if the user is assigned to the Struct Subadmin group. The system returns `1 (TRUE)` if the user is a member of the Struct Subadmin group. The system returns `0(FALSE)` if the user is not a member of the Struct Subadmin group, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

overlayFlag

A flag that indicates if the user is a member of an Overlay group and restricted to overlay and custom objects. The system returns `1 (TRUE)` if the user is a member of the Overlay group. The system returns `0 (FALSE)` if the user is not a member of the Overlay group, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

boverlayFlag

A flag that indicates if the user is restricted to editing base mode objects, unless also a member of an Overlay group. The system returns `1 (TRUE)` if the user is restricted to editing base mode objects. The system returns `0 (FALSE)` if the user is not restricted to base mode objects, is invalid, or is unknown. Specify `NULL` for this parameter if you do not want to retrieve this flag.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.148 ARWfdClearBreakpoint

Description

Removes the specified breakpoint from the server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdClearBreakpoint(
   ARControlStruct *control,
   unsigned int bpId,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

bpId

The ID of the breakpoint to be removed.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.149 ARWfdClearAllBreakpoints

Description

Removes all breakpoints from the server.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdClearAllBreakpoints(
   ARControlStruct *control,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.150 ARWfdExecute

Description

Instructs the debug server to begin execution.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdExecute(
   ARControlStruct *control,
   unsigned int mode,
   unsigned int *result,
   unsigned int *extraInfo1,
   unsigned int *extraInfo2,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

mode

The execution mode. You can specify `0` (single step) or `1` (run to breakpoint).

Return values

result

The result of the execute command.

extraInfo1

Other information associated with the result.

extraInfo2

Other information associated with the result.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.151 ARWfdGetCurrentLocation

Description

Requests the current location of the worker thread during debugging.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdGetCurrentLocation(
   ARControlStruct *control,
   unsigned int howFarBack,
   ARWfdCurrentLocation  *location,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

howFarBack

Stack location of a filter. A value of `0` requests information about the filter that is currently executing. Higher values request information about other positions on the call stack.

Return values

location

A current location structure. If the value of `howFarBack` exceeds the number of stack frames, the value of `location` will be `NULL`.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.152 ARWfdGetDebugMode

Description

Returns the current debug mode.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdGetDebugMode(
   ARControlStruct *control,
   unsigned int *mode,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

mode

The current debug mode.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.153 ARWfdGetFieldValues

Description

Requests the field-value list associated with the current schema at the current location.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdGetFieldValues(
   ARControlStruct *control,
   unsigned int howFarBack,
   ARFieldValueList *trFieldList,
   ARFieldValueList *dbFieldList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

howFarBack

Stack location of field value data.

Return values

trFieldList

A list of field/value pairs.

dbFieldList

A list of field/value pairs.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.154 ARWfdGetFilterQual

Description

Requests the field-value list associated with the current schema at the current location.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdGetFilterQual(
   ARControlStruct *control,
   ARQualifierStruct *filterQual,
   ARStatusList *status
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

filterQual

The filter qualifier structure for the filter that is currently executing.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.155 ARWfdGetKeywordValue

Description

Retrieves the value of a keyword, if possible.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdGetKeywordValue(
   ARControlStruct *control,
   unsigned int keywordId,
   ARValueStruct *value,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

keywordId

The ID representing the keyword.

Return values

value

The value of the keyword.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.156 ARWfdGetUserContext

Description

Retrieves information associated with the workflow user.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdGetUserContext(
   ARControlStruct *control,
   unsigned int mask,
   ARWfdUserContext *userInfo,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

mask

Mask for individual items.

Return values

userInfo

User details.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.157 ARWfdListBreakpoints

Description

Returns a list of server breakpoints.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdListBreakpoints(
   ARControlStruct *control,
   ARWfdRmtBreakpointList *bpList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

Return values

bpList

A list of server breakpoints.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.158 ARWfdSetBreakpoint

Description

Sets a breakpoint on the server, and overwrites if needed.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdSetBreakpoint(
   ARControlStruct *control,
   ARWfdRmtBreakpoint *inBp,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

inBp

The new breakpoint to set.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.159 ARWfdSetDebugMode

Description

Sets a new debug mode.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdSetDebugMode(
   ARControlStruct *control,
   unsigned int mode,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

mode

The new debug mode.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.160 ARWfdSetFieldValues

Description

Overwrites the field-value list associated with the current schema at the current location.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdSetFieldValues(
   ARControlStruct *control,
   ARFieldValueList *trFieldList,
   ARFieldValueList *dbFieldList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

trFieldList

A list of field/value pairs.

dbFieldList

A list of field/value pairs.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.161 ARWfdSetQualifierResult

Description

Forces the qualifier result to the specified Boolean value.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdSetQualifierResult(
   ARControlStruct *control,
   ARBoolean result,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

result

If `result` is TRUE, workflow execution will take the **if** path; if `result` is FALSE, workflow execution will take the **else** path. This occurs regardless of the actual value returned by evaluation of the filter qualifier.

This entry point must be called before the filter qualifier is evaluated, or there will be no effect.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.162 ARWfdTerminateAPI

Description

Causes workflow to return with an optionally specified error at the next opportunity. If an error is not specified, a generic `TERMINATED_BY_DEBUGGER` error will be returned.

Privileges

BMC Remedy AR System administrator.

Synopsis

```
#include "ar.h"
#include "arerrno.h"
#include "arextern.h"
#include "arstruct.h"

int ARWfdTerminateAPI(
   ARControlStruct *control,
   unsigned int errorCode,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user** and **server** fields are required.

errorCode

The return code for the API call.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

6.2.163 FreeAR

Description

Recursively free all allocated memory associated with a particular BMC Remedy AR System data structure. All structure components must be in allocated memory to use these functions.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arfree.h"

void FreeARAccessNameList(
   ARAccessNameList *value,
   ARBoolean freeStruct)

void FreeARAccessNamePtrList(
   ARAccessNamePtrList *value,
   ARBoolean freeStruct)

void FreeARActiveLinkActionList(
   ARActiveLinkActionListList *value,
   ARBoolean freeStruct)

void FreeARActiveLinkActionListList(
   ARActiveLinkActionList *value,
   ARBoolean freeStruct)

void FreeARActiveLinkActionStruct(
   ARActiveLinkActionStruct *value,
   ARBoolean freeStruct)

void FreeARActiveLinkSvcActionStruct(
   ARActiveLinkSvcActionStruct *value,
   ARBoolean freeStruct)

void FreeARArchiveInfoList(
   ARArchiveInfoList *value,
   ARBoolean freeStruct)

void FreeARArchiveInfoStruct(
   ARArchiveInfoStruct *value,
   ARBoolean freeStruct)

void FreeARArithOpAssignStruct(
   ARArithOpAssignStruct *value,
   ARBoolean freeStruct)

void FreeARArithOpStruct(
   ARArithOpStruct *value,
   ARBoolean freeStruct)

void FreeARAssignFieldStruct(
   ARAssignFieldStruct *value,
   ARBoolean freeStruct)

void FreeARAssignSQLStruct(
   ARAssignSQLStruct *value,
   ARBoolean freeStruct)

void FreeARAssignStruct(
   ARAssignStruct *value,
   ARBoolean freeStruct)

void FreeARAttachStruct(
   ARAttachStruct *value,
   ARBoolean freeStruct)

void FreeARAuditInfoList(
   ARAuditInfoList *value,
   ARBoolean freeStruct)

void FreeARAuditInfoStruct(
   ARAuditInfoStruct *value,
   ARBoolean freeStruct)

void FreeARAutomationStruct(
   ARAutomationStruct *value,
   ARBoolean freeStruct)

void FreeARBooleanList(
   ARBooleanList *value,
   ARBoolean freeStruct)

void FreeARBooleanListList(
   ARBooleanListList *value,
   ARBoolean freeStruct)

void FreeARBooleanMatrix(
   ARBooleanMatrix *value,
   ARBoolean freeStruct)

void FreeARBufStruct(
   ARBufStruct *value,
   ARBoolean freeStruct)

void FreeARBulkEntryReturn(
   ARBulkEntryReturn *value,
   ARBoolean freeStruct)

void FreeARBulkEntryReturnList(
   ARBulkEntryReturnList *value,
   ARBoolean freeStruct)

void FreeARByteList(
   ARByteList *value,
   ARBoolean freeStruct)

void FreeARCallGuideStruct(
   ARCallGuideStruct *value,
   ARBoolean freeStruct)

void FreeARCharMenuItemStruct(
   ARCharMenuItemStruct *value,
   ARBoolean freeStruct)
```

```
void FreeARCharMenuList(
   ARCharMenuList *value,
   ARBoolean freeStruct)

void FreeARCharMenuSSStruct(
   ARCharMenuSSStruct *value,
   ARBoolean freeStruct)

void FreeARCharMenuStruct(
   ARCharMenuStruct *value,
   ARBoolean freeStruct)

void FreeARCharMenuStructList(
   ARCharMenuStructList *value,
   ARBoolean freeStruct)

void FreeARCOMMethodList(
   ARCOMMethodList *value,
   ARBoolean freeStruct)

void FreeARCOMMethodParmList(
   ARCOMMethodParmList *value,
   ARBoolean freeStruct)

void FreeARCOMMethodParmStruct(
   ARCOMMethodParmStruct *value,
   ARBoolean freeStruct)

void FreeARCOMMethodStruct(
   ARCOMMethodStruct *value,
   ARBoolean freeStruct)

void FreeARComplexEntryService(
   ARComplexEntryService *value,
   ARBoolean freeStruct)

void FreeARComplexEntryServiceList(
   ARComplexEntryServiceList *value,
   ARBoolean freeStruct)

void FreeARComplexEntryServiceOut(
   ARComplexEntryServiceOut *value,
   ARBoolean freeStruct)

void FreeARComplexEntryServiceOutList(
   ARComplexEntryServiceOutList *value,
   ARBoolean freeStruct)

void FreeARCompoundSchema(
   ARCompoundSchema *value,
   ARBoolean freeStruct)

void FreeARCompoundSchemaList(
   ARCompoundSchemaList *value,
   ARBoolean freeStruct)

void FreeARCOMValueStruct(
   ARCOMValueStruct *value,
   ARBoolean freeStruct)

void FreeARContainerInfoList(
   ARContainerInfoList *value,
   ARBoolean freeStruct)

void FreeARContainerOwnerObjList(
   ARContainerOwnerObjList *value,
   ARBoolean freeStruct)

void FreeARContainerOwnerObjListList(
   ARContainerOwnerObjListList *value,
   ARBoolean freeStruct)

void FreeARContainerTypeList(
   ARContainerTypeList *value,
   ARBoolean freeStruct)

void FreeARCoordList(
   ARCoordList *value,
   ARBoolean freeStruct)

void FreeARCurrencyList(
   ARCurrencyList *value,
   ARBoolean freeStruct)

void FreeARCurrencyStruct(
   ARCurrencyStruct *value,
   ARBoolean freeStruct)

void FreeARDataMappingInfoList(
   ARDataMappingInfoList *value,
   ARBoolean freeStruct)

void FreeARDataMappingInfoStruct(
   ARDataMappingInfoStruct *value,
   ARBoolean freeStruct)

void FreeARDDEStruct(
   ARDDEStruct *value,
   ARBoolean freeStruct)

void FreeARDiaryList(
   ARDiaryList *value,
   ARBoolean freeStruct)

void FreeARDisplayInstanceList(
   ARDisplayInstanceList *value,
   ARBoolean freeStruct)

void FreeARDisplayInstanceListList(
   ARDisplayInstanceListList *value,
   ARBoolean freeStruct)

void FreeARDisplayInstanceListPtrList(
   ARDisplayInstanceListPtrList *value,
   ARBoolean freeStruct)
```

```
void FreeARDisplayInstanceStruct(
   ARDisplayInstanceStruct *value,
   ARBoolean freeStruct)

void FreeARDisplayList(
   ARDisplayList *value,
   ARBoolean freeStruct)

void FreeAREntryBlockList(
   AREntryBlockList *value,
   ARBoolean freeStruct)

void FreeAREntryBlockStruct(
   AREntryBlockStruct *value,
   ARBoolean freeStruct)

void FreeAREntryIdList(
   AREntryIdList *value,
   ARBoolean freeStruct)

void FreeAREntryIdListList(
   AREntryIdListList *value,
   ARBoolean freeStruct)

void FreeAREntryInfoList(
   AREntryInfoListList *value,
   ARBoolean freeStruct)

void FreeAREntryListFieldList(
   AREntryListFieldList *value,
   ARBoolean freeStruct)

void FreeAREntryListFieldListList(
   AREntryListFieldListList *value,
   ARBoolean freeStruct)

void FreeAREntryListFieldValueList(
   AREntryListFieldValueList *value,
   ARBoolean freeStruct)

void FreeAREntryListFieldValueStruct(
   AREntryListFieldValueStruct *value,
   ARBoolean freeStruct)

void FreeAREntryListList(
   AREntryListList *value,
   ARBoolean freeStruct)

void FreeAREnumItemList(
   AREnumItemList *value,
   ARBoolean freeStruct)

void FreeARFieldAssignList(
   ARFieldAssignList *value,
   ARBoolean freeStruct)

void FreeARFieldAssignStruct(
   ARFieldAssignStruct *value,
   ARBoolean freeStruct)

void FreeARFieldInfoList(
   AREntryFieldInfoList *value,
   ARBoolean freeStruct)

void FreeARFieldInfoStruct(
   AREntryFieldInfoStruct *value,
   ARBoolean freeStruct)

void FreeARFieldLimitList(
   ARFieldLimitList *value,
   ARBoolean freeStruct)

void FreeARFieldLimitPtrList(
   ARFieldLimitPtrList *value,
   ARBoolean freeStruct)

void FreeARFieldLimitStruct(
   ARFieldLimitStruct *value,
   ARBoolean freeStruct)

void FreeARFieldMappingList(
   ARFieldFieldMappingList *value,
   ARBoolean freeStruct)

void FreeARFieldMappingPtrList(
   ARFieldMappingPtrList *value,
   ARBoolean freeStruct)

void FreeARFieldValueList(
   ARFieldValueList *value,
   ARBoolean freeStruct)

void FreeARFieldValueListList(
   ARFieldValueListList *value,
   ARBoolean freeStruct)

void FreeARFieldValueOrArithStruct(
   ARFieldValueOrArithStruct *value,
   ARBoolean freeStruct)

void FreeARFieldValueStruct(
   ARFieldValueStruct *value,
   ARBoolean freeStruct)

void FreeARFilterActionList(
   ARFilterActionList *value,
   ARBoolean freeStruct)

void FreeARFilterActionListList(
   ARFilterActionListList *value,
   ARBoolean freeStruct)

void FreeARFilterActionNotifyAdvanced(
   ARFilterActionNotifyAdvanced *value,
   ARBoolean freeStruct)
```

```
void FreeARFilterActionStruct(
   ARFilterActionStruct *value,
   ARBoolean freeStruct)

void FreeARFullTextInfoList(
   ARFullTextInfoList *value,
   ARBoolean freeStruct)

void FreeARFullTextInfoRequestList(
   ARFullTextInfoRequestList *value,
   ARBoolean freeStruct)

void FreeARFunctionAssignStruct(
   ARFunctionAssignStruct *value,
   ARBoolean freeStruct)

void FreeARGotoGuideLabelStruct(
   ARGotoGuideLabelStruct( *value,
   ARBoolean freeStruct)

void FreeARGroupInfoList(
   ARGroupInfoList *value,
   ARBoolean freeStruct)

void FreeARHostIDTypeList(
   ARHostIDTypeList *value,
   ARBoolean freeStruct)

void FreeARIndexList(
   ARIndexList *value,
   ARBoolean freeStruct)

void FreeARIndexListList(
   ARIndexListList *value,
   ARBoolean freeStruct)

void FreeARInternalIdList(
   ARInternalIdList *value,
   ARBoolean freeStruct)

void FreeARInternalIdListList(
   ARInternalIdListList *value,
   ARBoolean freeStruct)

void FreeARLicenseInfoList(
   ARLicenseInfoList *value,
   ARBoolean freeStruct)

void FreeARLicenseInfoStruct(
   ARLicenseInfoStruct *value,
   ARBoolean freeStruct)

void FreeARLicenseNameStruct(
   ARLicenseNameStruct *value,
   ARBoolean freeStruct)

void FreeARLicenseValidList(
   ARLicenseValidList *value,
   ARBoolean freeStruct)

void FreeARLicenseValidStruct(
   ARLicenseValidStruct *value,
   ARBoolean freeStruct)

void FreeARLocaleList(
   ARLocaleList *value,
   ARBoolean freeStruct)

void FreeARLocStruct(
   ARLocStruct *value,
   ARBoolean freeStruct)

void FreeARMacroParmList(
   FreeARMacroParmList *value,
   ARBoolean freeStruct)

void FreeARNameList(
   ARNameList *value,
   ARBoolean freeStruct)

void FreeARNamePtrList(
   ARNamePtrList *value,
   ARBoolean freeStruct)

void FreeARObjectInfoList(
   ARObjectInfoList *value,
   ARBoolean freeStruct)

void FreeARObjectInfoStruct(
   ARObjectInfoStruct *value,
   ARBoolean freeStruct)

void FreeAROpenDlgStruct(
   AROpenDlgStruct *value,
   ARBoolean freeStruct)

void FreeARPasswordList(
   ARPasswordList *value,
   ARBoolean freeStruct)

void FreeARPermissionList(
   ARPermissionList *value,
   ARBoolean freeStruct)

void FreeARPermissionListList(
   ARPermissionListList *value,
   ARBoolean freeStruct)

void FreeARPermissionListPtrList(
   ARPermissionListPtrList *value,
   ARBoolean freeStruct)

void FreeARPropList(
   ARPropList *value,
   ARBoolean freeStruct)
```

```
void FreeARPropListList(
   ARPropListList *value,
   ARBoolean freeStruct)

void FreeARPropStruct(
   ARPropStruct *value,
   ARBoolean freeStruct)

void FreeARPushFieldsList(
   ARPushFieldsList *value,
   ARBoolean freeStruct)

void FreeARQualifierList(
   ARQualifierList *value,
   ARBoolean freeStruct)

void FreeARQualifierStruct(
   ARQualifierStruct *value,
   ARBoolean freeStruct)

void FreeARReferenceList(
   ARReferenceList *value,
   ARBoolean freeStruct)

void FreeARReferenceListList(
   ARReferenceListList *value,
   ARBoolean freeStruct)

void FreeARReferenceStruct(
   ARReferenceStruct *value,
   ARBoolean freeStruct)

void FreeARReferenceTypeList(
   ARReferenceTypeList *value,
   ARBoolean freeStruct)

void FreeARRelOpStruct(
   ARRelOpStruct *value,
   ARBoolean freeStruct)

void FreeARRoleInfoList(
   ARRoleInfoList *value,
   ARBoolean freeStruct)

void FreeARSchemaInheritanceList(
   ARSchemaInheritanceList *value,
   ARBoolean freeStruct)

void FreeARSchemaInheritanceListList(
   ARSchemaInheritanceListList *value,
   ARBoolean freeStruct)

void FreeARSchemaInheritanceStruct(
   ARSchemaInheritanceStruct  *value,
   ARBoolean freeStruct)

void FreeARServerInfoList(
   ARServerInfoList *value,
   ARBoolean freeStruct)

void FreeARServerInfoRequestList(
   ARServerInfoRequestList *value,
   ARBoolean freeStruct)

void FreeARServerNameList(
   ARServerNameList *value,
   ARBoolean freeStruct)

void FreeARSignalList(
   ARSignalList *value,
   ARBoolean freeStruct)

void FreeARSortList(
   ARSortList *value,
   ARBoolean freeStruct)

void FreeARSortListList(
   ARSortListList *value,
   ARBoolean freeStruct)

void FreeARSQLStruct(
   ARSQLStruct *value,
   ARBoolean freeStruct)

void FreeARStatisticsResultList(
   ARStatisticsResultList *value,
   ARBoolean freeStruct)

void FreeARStatusHistoryList(
   ARStatusHistoryList *value,
   ARBoolean freeStruct)

void FreeARStatusList(
   ARStatusList *value,
   ARBoolean freeStruct)

void FreeARStatusListList(
   ARStatusListList *value,
   ARBoolean freeStruct)

void FreeARStructItemList(
   ARStructItemList *value,
   ARBoolean freeStruct)

void FreeARSupportFileInfoList(
   ARSupportFileInfoList *value,
   ARBoolean freeStruct)

void FreeARSupportFileInfoStruct(
   ARSupportFileInfoStruct *value,
   ARBoolean freeStruct)

void FreeARTextStringList(
   ARTextStringList *value,
   ARBoolean freeStruct)
```

```
void FreeARTimestampList(
   ARTimestampList *value,
   ARBoolean freeStruct)

void FreeARUnsignedIntList(
   ARUnsignedIntList *value,
   ARBoolean freeStruct)

void FreeARUnsignedIntPtrList(
   ARUnsignedIntPtrList *value,
   ARBoolean freeStruct)

void FreeARUserInfoList(
   ARUserInfoList *value,
   ARBoolean freeStruct)

void FreeARUserLicenseList(
   ARUserLicenseList *value,
   ARBoolean freeStruct)

void FreeARUserLicenseStruct(
   ARUserLicenseStruct *value,
   ARBoolean freeStruct)

void FreeARValueList(
   ARValueList *value,
   ARBoolean freeStruct)

void FreeARValueListList(
   ARValueListList *value,
   ARBoolean freeStruct)

void FreeARValuePtrList(
   ARValuePtrList *value,
   ARBoolean freeStruct)

void FreeARValueStruct(
   ARValueStruct *value,
   ARBoolean freeStruct)

void FreeARVuiInfoList(
   ARVuiInfoList *value,
   ARBoolean freeStruct)

void FreeARVuiInfoStruct(
   ARVuiInfoStruct *value,
   ARBoolean freeStruct)

void FreeARWaitStruct(
   ARWaitStruct *value,
   ARBoolean freeStruct)

void FreeARWorkflowConnectList(
   ARWorkflowConnectList *value,
   ARBoolean freeStruct)

void FreeARWorkflowConnectStruct(
   ARWorkflowConnectStruct *value,
   ARBoolean freeStruct)

void FreeARXMLInputDoc(
   ARXMLInputDoc *value,
   ARBoolean freeStruct)

void FreeARXMLOutputDoc(
   ARXMLOutputDoc *value,
   ARBoolean freeStruct)

void FreeARXMLParsedStream(
   ARXMLParsedStream *value,
   ARBoolean freeStruct)

void FreeARXMLParserHandle(
   ARXMLParserHandle *value,
   ARBoolean freeStruct)

void FreeListARCharMenuStruct(
   ListARCharMenuStruct *value,
   ARBoolean freeStruct)
```

Input arguments

value

A pointer to the structure that you want to free. The system does not perform an operation if the structure is a list with zero items or you specify `NULL` for this parameter.

freeStruct

A flag that indicates whether you need to free the top-level structure. If you allocated memory for the top-level structure, specify `1` (`TRUE`) to free both the structure and its contents. If you used a stack variable for the top-level structure, specify `0` (`FALSE`) to free only the contents of the structure.

6.2.164 ARBeginClientManagedTransaction

Description

Enables you to start a client-managed database transaction. This function returns the transaction handle that other control records can use and allows both BMC Remedy AR System and CMDB calls to be made in the same transaction.

Before using client-managed transaction functions, you can set the maximum number of client transactions and the transaction timeout parameters in either of the following ways:

- Update the **Maximum Concurrent Transactions** and **Transaction Timeout** fields on the Advanced tab of the Server Information form. (See Setting security options using Server information form settings.)

- Set the Client-Managed-Transaction-Timeout and Maximum-Concurrent-Client-Managed-Transactions options in the BMC Remedy AR System configuration file (**ar.cfg** or **ar.conf**).

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arextern.h"
int ARBeginClientManagedTransaction(
    ARControlStruct        *control,
    char            *transactionHandle,
    ARStatusList        *status)
```

Input arguments

control

The control record for the operation, which contains information about the user requesting the operation and the server on which the operation is to be performed.

Return values

transactionHandle

The transaction handle ID of the database transaction.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

AREndClientManagedTransaction

6.2.165 AREndClientManagedTransaction

Description

Enables you to end the client-managed database transaction. Depending on the transaction option used, the transaction is either rolled back or committed.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arextern.h"
int AREndClientManagedTransaction(
    ARControlStruct        *control,
    int            *transactionOption,
    ARStatusList        *status)
```

Input arguments

control

The control record for the operation, which contains information about the user requesting the operation and the server on which the operation is to be performed.

transactionOption

The transaction option, which allows you to rollback or commit the transaction. The possible transaction option values are `AR_ROLLBACK_CLIENT_MANAGED_TRANSACTION` to roll back the transaction and `AR_COMMIT_CLIENT_MANAGED_TRANSACTION` to commit the transaction.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARBeginClientManagedTransaction

6.2.166 ARSetClientManagedTransaction

Description

Enables you to set a client-managed database transaction handle to the given user context. After the transaction handle is set, all subsequent entry calls will be in the transaction that is associated with this handle. With this function, you can make data calls on different user contexts in the same transaction. You can also use handles received from the equivalent CMDB call.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arextern.h"
int ARSetClientManagedTransaction (
    ARControlStruct        *control,
    int            *transactionHandle,
    ARStatusList        *status)
```

Input arguments

control

The control record for the operation, which contains information about the user requesting the operation and the server on which the operation is to be performed.

transactionHandle

The transaction handle ID of the database transaction.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARRemoveClientManagedTransaction

6.2.167 ARRemoveClientManagedTransaction

Description

Enables you to remove the client-managed database transaction. All the subsequent calls for this control record will not be part of the transaction associated with the transaction handle. You can also use handles received from the equivalent CMDB call.

Privileges

All users.

Synopsis

```
#include "ar.h"
#include "arextern.h"
```

```
int ARRemoveClientManagedTransaction (
    ARControlStruct        *control,
    ARStatusList        *status)
```

Input arguments

control

The control record for the operation, which contains information about the user requesting the operation and the server on which the operation is to be performed.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetClientManagedTransaction

7 Creating and executing BMC Remedy AR System C API programs

Use the following information to understand how to initialize and shut down the system and how to perform other common tasks related to the BMC Remedy AR System C API programs.

7.1 Program structure

The BMC Remedy AR System C API programs consist of the following sections:

Startup/Initialization	Call ARInitialization to perform server- and network-specific initialization operations for connecting to the BMC Remedy 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 (or use the free function) to free all allocated memory associated with a specific data structure (see Freeing allocated memory).
Shutdown/Cleanup	Call to perform environment-specific cleanup routines and disconnect from the BMC Remedy 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.

The following code fragment provides an example of the generic program structure. Because almost every BMC Remedy AR System C API function has a **control** parameter as its first input argument, the template includes code for loading the **ARControlStruct** structure (see Login and session information).

Note

ARInitialization does not use login information from the **control** structure, so you can load it into the control structure either before or after you issue the **ARInitialization** call. In the example, it is done after.

```
{
ARControlStruct   control;
ARStatusList      status;
int            rtn;
    ...
    ...

/****************************************************************/
/* Perform Startup                                              */
/****************************************************************/
if (ARInitialization(&control, &status) >= AR_RETURN_ERROR){
    printf("\n **** initialization error ****\n");
    PrintARStatusList(&status);
    exit(3);
}
FreeARStatusList(&status, FALSE);

/****************************************************************/
/* Load ARControlStruct                                         */
/****************************************************************/
strncpy(control.user, "Demo", AR_MAX_ACCESS_NAME_SIZE);
                                               /* use Demo user */
control.user[AR_MAX_ACCESS_NAME_SIZE] = '\0';
strcpy(control.password, "");           /* could load from file */
memset(&control.localeinfo, 0, sizeof(ARLocalizationInfo));
                                          /* use default locale */
strncpy(control.server, argv[1], AR_MAX_SERVER_SIZE);
control.server[AR_MAX_SERVER_SIZE] = '\0';
strcpy(control.authString, "");         /* could load from file */


/****************************************************************/
/* Perform System Work                                          */
/****************************************************************/
rtn = ARGetEntry(&control, ..., &status);
if( rtn ... )
    ...
    ...

/****************************************************************/
/* Perform Cleanup                                              */
/****************************************************************/
(&control, &status);
FreeARStatusList(&status, FALSE);
```

7.2 Performing common tasks

Each object in the BMC Remedy AR System has an `ARGetList` *object* function that returns a list of records that match the specified criteria. Depending on the object, this list contains zero or more names or IDs (or both) that uniquely identify the record. The following table shows the unique identifier for each BMC Remedy AR System object.

BMC Remedy AR System object	Return parameter	Parameter type	Unique identifier
Active link	nameList	ARNameList *	Name
Character menu	nameList	ARNameList *	Name
Container	conList	ARContainerInfoList *	Name
Entry	entryList	AREntryListList *	ID
Escalation	nameList	ARNameList *	Name
Field	idList	ARInternalIdList *	ID
Filter	nameList	ARNameList *	Name
Group	groupList	ARGroupInfoList *	ID
Image	imageList	ARNameList *	Name
Schema	nameList	ARNameList *	Name
Server	serverList	ARServerNameList *	Name
User	userList	ARUserInfoList *	Name
View (VUI)	idList	ARInternalIdList *	ID

These names or IDs can then be passed as input arguments to the `ARGet` *object*, `ARSet` *object*, or `ARDelete` *object* functions to perform the corresponding operation on those records.

Unqualified `ARGetList` *Object* calls generate a complete list of records (for a particular object). You can select a subset of these records by using a qualification. When retrieving a list of entries, you can specify any set of selection conditions by using `ARQualifierStruct` (see Representing qualifications with structures). For other objects, you can retrieve lists limited to records modified after a specified date and time value. For fields, views, filters, escalations, and active links, you can also limit it to items associated with a specified schema.

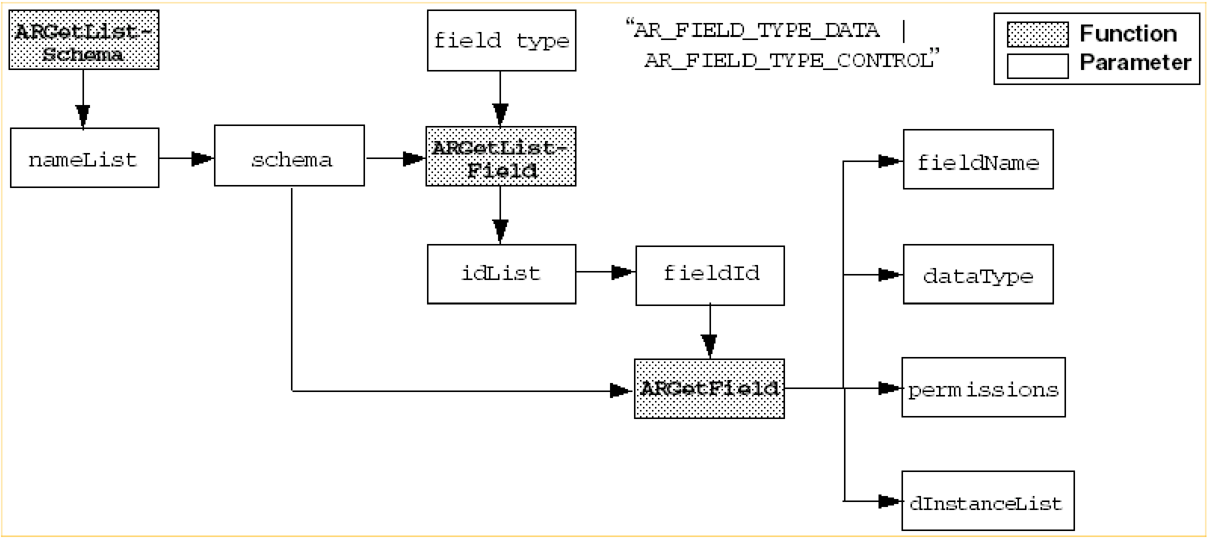
The following examples use common tasks to illustrate these concepts. Each of the first two examples include a brief task description, an outline of the conceptual steps involved, and a diagram of the C API functions and parameters used to accomplish the task. These examples also demonstrate the general problem-solving approach of breaking tasks into conceptual steps. This approach might help you identify the sequence of C API functions needed to solve your programming problem. The third example consists of a small sample program that retrieves and prints a list of available BMC Remedy AR System servers.

7.2.1 Example 1: Retrieving field properties

The goal in this example is to retrieve selected properties for a particular schema field. In this case, the field is a data or control field, and the properties shown are the field name, data type, access control information, and display properties. The following high-level steps are illustrated in the following figure.

7.2.2 To retrieve selected properties for a field

1. Retrieve a list of available schemas (`ARGetListSchema`), and select the desired schema name (**schema**) from the list (**nameList**).
2. Retrieve a list of data, trim, and control fields (**fieldType**) associated with the schema (`ARGetListField`).
3. Select the desired field (**fieldId**) from the list (**idList**), and retrieve the specified properties for that field (`ARGetField`).
   Retrieving selected properties from a field
   (Click the image to expand it.)



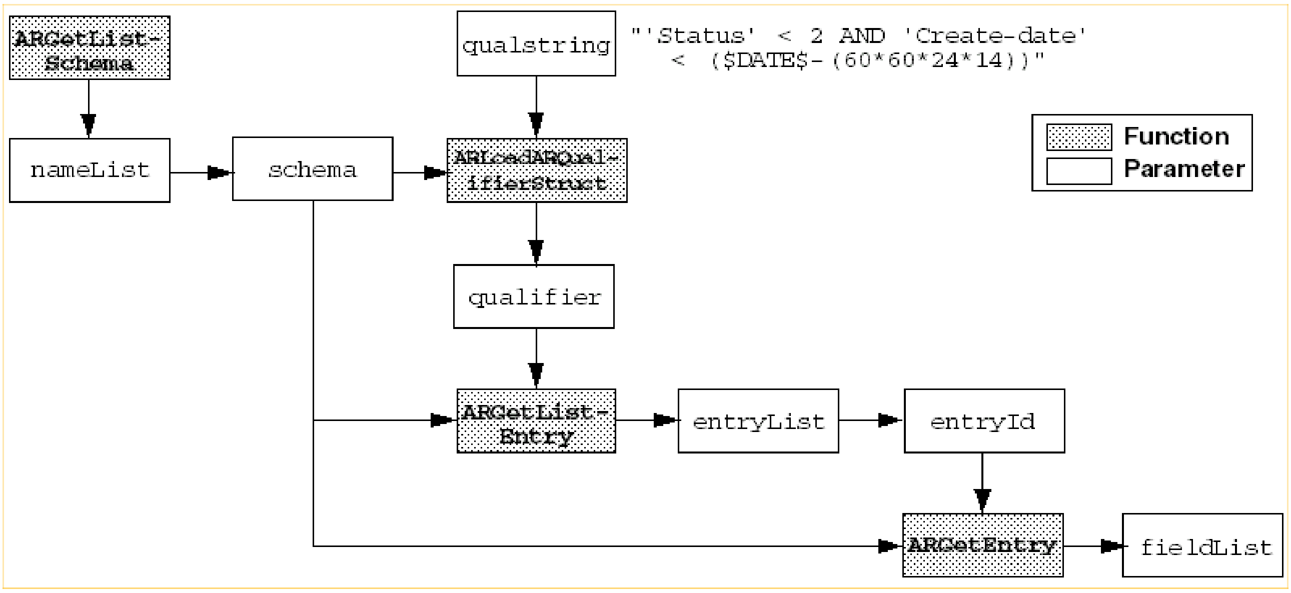
7.2.3 Example 2: Retrieving selected entries

The goal in this example is to retrieve the set of schema entries that match specified query conditions. The selection criteria are those entries that are either New or Open and were created more than two weeks ago. The following high-level steps are illustrated in the following figure.

7.2.4 To retrieve selected entries from a schema

1. Retrieve a list of available schemas (`ARGetListSchema`), and select the desired schema name (**schema**) from the list (**nameList**).
2. Create a string with the desired query conditions (**qualString**), and place it into the proper structure (`ARLoadARQualifierStruct`).
3. Retrieve a list of entries (`ARGetListEntry`) from the schema that match the query conditions (**qualifier**).
4. If desired, select a particular entry (**entryId**) from the list (**entryList**), and retrieve the field data (**fieldList**) for that entry (`ARGetEntry`).

Retrieving the field data
(Click the image to expand it.)



7.2.5 Example 3: Retrieving a schema list

```
#include "ar.h"
#include "arextern.h"
#include "arfree.h"

void printStatusList(ARStatusList *theList)
{
    unsigned int i;
    for (i = 0; i < theList->numItems; i++)

    {
        if (theList->statusList[i].appendedText == NULL ||
                theList->statusList[i].appendedText[0] == '\0')
            printf("%s", theList->statusList[i].messageText);
        else
            printf("%s : %s", theList->statusList[i].messageText,
                    theList->statusList[i].appendedText);
        printf(" (ARERR %d)\n", theList->statusList[i].messageNum);
    }
}

int main(void)
{
    ARControlStruct control = {0,0,"Demo", "", "", 0, "",
                "yourservername"};
    ARNameList formNameList;
    unsigned int i;
    ARStatusList status = {0, NULL};
    if (ARInitialization(&control, &status) >= AR_RETURN_ERROR)
    {
        printStatusList(&status);
        FreeARStatusList(&status, FALSE);
        return 1;
    }
    FreeARStatusList(&status, FALSE);

    if (ARGetListSchema(&control, 0,
                AR_LIST_SCHEMA_ALL |AR_HIDDEN_INCREMENT,
                NULL, NULL, NULL,
                &formNameList, &status) >= AR_RETURN_ERROR)
            printStatusList(&status);
    else
    {
        for (i = 0; i < formNameList.numItems; i++)
                printf("%s\n", formNameList.nameList[i]);
    }

    FreeARStatusList(&status, FALSE);
    FreeARNameList(&formNameList, FALSE);
    if ((&control, &status) >= AR_RETURN_ERROR)
        printStatusList(&status);
    FreeARStatusList(&status, FALSE);
    return 0;
}
```

7.3 Specifying fields or keywords

The BMC Remedy AR System includes several actions that involve text:

- Sending a notification (filters and escalations)
- Sending a message (filters and active links)
- Running a process (filters, escalations, and active links)
- Specifying a macro parameter (active links)

As BMC Remedy Developer Studio runs in a localized environment, you can include a field or keyword value in a character string by specifying the name of the desired field or keyword.

A C API program, however, can operate in multiple environments. Because field names and keywords vary across environments, you must specify the field ID or keyword code (which are environment-independent) to include one of these values in a character string. The following table shows the syntax for specifying a field or keyword value.

Field	$ *id* $	ID of the desired field (for example, 7)
Keyword	$ *code* $	Code associated with the desired keyword in the **ar.h** file (for example, -5 for the current schema). Use -1 to specify NULL. A hyphen is required to distinguish a keyword code from a field ID.

Note

Field IDs and keyword codes are not required when specifying a qualification string for the **ARLoadARQualifierStruct** function because it automatically translates all values for you.

7.4 Error checking

All the C API functions (except `FreeAR`) return a status parameter (`ARStatusList`) that consists of zero or more notes, warnings, or errors generated from the function call (see Status information).

Each message consists of a value that indicates the type of message, the message number, and the message text (in the language specified in the control structure). More serious errors are listed first with lesser warnings and informational messages following. Within each category, messages are listed in reverse chronological order, with the most recent message first.

The return of the function itself is an integer value that indicates the success or failure of the call (see the following table). This value represents the status associated with the first (most recent and most serious) message in the list.

`0`	`AR_RETURN_OK`	Operation successful-- status might contain one or more *informational* messages.
`1`	`AR_RETURN_WARNING`	Operation successful, but a problem was encountered-- status contains one or more *warning* messages and might also contain informational messages.
`2`	`AR_RETURN_ERROR`	Operation failed-- status contains one or more *error* messages and might also contain warning or informational messages.
`3`	`AR_RETURN_FATAL`	Operation failed-- status might contain one or more messages.
`4`	`AR_RETURN_BAD_STATUS`	Invalid status parameter--operation canceled.

You can ignore warnings and informational messages, although reporting warnings is often helpful. Because the C API returns all errors in a common structure, you can perform all error handling by using a single, generic routine. The following example provides a sample routine for checking return values.

```
ARStatusListstatus;
char            apicall[ar_MAX_APICALL_SIZE]; /* function name*/
int            rtn; /* function return value*/

rtn = ARGetEntry(&control, ..., &status);
strcpy(apicall,"ARGetEntry");
switch( rtn ){
    case AR_RETURN_OK:
        printf("\t%s: Successful\n", apicall);
        PrintARStatusList( &status );
        break;
    case AR_RETURN_WARNING:
        printf("\t%s: Warning\n", apicall);
        PrintARStatusList( &status );
```

```
            break;
        case AR_RETURN_ERROR:
            printf("\t%s: Error\n", apicall);
            PrintARStatusList( &status );
            exit(3);
            break;
        case AR_RETURN_FATAL:
            printf("\t%s: Fatal\n", apicall);
            exit(3);
            break;
        case AR_RETURN_BAD_STATUS:
            printf("\t%s: Bad Status\n", apicall);
            exit(3);
            break;
        default:
            printf("\t%s: Invalid return value: %d\n", apicall, rtn);
            exit(3);
    }
```

Because the `status` structure uses allocated memory, you must free that memory after every C API call by using `FreeARStatusList`. You can call these functions regardless of whether the `status` structure contains any messages, because they perform no action if there are no messages to free (for more information, see Freeing allocated memory).

7.5 Executing C API programs in workflow

The most obvious way to execute a C API program is from the command line. You can execute a C API program directly, integrate it with other processes or commands in a shell script, or schedule it for regular execution by using a UNIX **cron** job or the Windows Scheduler.

The second way to execute a C API program is through workflow. You can call a C API program in a filter, escalation, or active link to perform the Set Fields, Push Fields, or Run Process actions.

7.5.1 Set Fields and Push Fields

The Set Fields and Push Fields actions assign a specified value to a particular schema field. You can assign a constant, a value from another field, or an operation result value (see Set Fields action and structures or Push Fields action and structures). To assign the result of an operating system process or command (including a C API program), specify the following field value:

```
$PROCESS$ APIProgram parameters
```

The Set Fields and Push Fields operations block the BMC Remedy AR System server (the server waits for the operation to complete before performing any other actions). If the Set Fields or Push Fields action calls a C API program that executes on the same BMC Remedy AR System server, a deadlock could occur if there are not multiple server threads available to handle calls. When only a single server thread is configured, the server cannot satisfy the request from the API program until it completes the current operation, but the server cannot complete the operation until it executes the API program.

This deadlock causes a time-out, and the Set Fields or Push Fields operation fails. To avoid this problem when running your API program against the same server, configure your server to have multiple fast and list server threads. See Defining queues and configuring threads.

Note

The default process time-out interval is five seconds. You can adjust this setting by using the BMC Remedy AR System Administration: Server Information form. Valid values are from 1 to 60 seconds.

7.5.2 Run Process

The Run Process action executes the specified operating system process or command. To execute a C API program, specify the following as the command value:

```
API Program parameters
```

Unlike a Set Fields operation, the Run Process action does *not* block the BMC Remedy AR System server. The system places these tasks in the Run Process queue, where they are deferred until the server has completed all database transactions.

If the specified process requires a field value (for example, sending a pager message to the user to whom a particular ticket has been assigned), the server might use different values depending on the type of database operation and whether it was successful. The following table summarizes the possible scenarios.

Database transaction	Submit	Modify
Succeeds	Uses *new* database value	Uses *new* database value
Fails	Action not performed	Uses *old* database value

If the specified process does not require a field value (for example, sending a pager message to a particular user), the server does not execute the queued processes after an unsuccessful database transaction.

7.6 Program design tips

- Use industry-standard tools.
  - Use a C or C++ compiler adhering to the ANSI standard. If you are using the API under Windows, use Microsoft Visual C++ (version 6.0 or later).
  - If the program must run on both UNIX and Windows, use POSIX routines for system calls.
- Use variables (instead of hard coding) for more flexibility and less maintenance.
  - Use symbolic constants (**#define** values from **ar.h**).
  - Use environment variables.

- Use files to store menu or selection values instead of writing them into your program.
- Minimize network traffic.
- Use bulk transfer to retrieve data and to cache the information locally. Omit unnecessary fields when retrieving data.

7.7 Multithreaded C API clients

The BMC Remedy AR System C API supports multithreaded clients through the use of sessions. Each session maintains its own state information, enabling simultaneous operations against BMC Remedy 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.

7.8 Impersonating a user

The `ARSetImpersonatedUser` API call enables you to set the current user to be any other valid user while still running as **aradmin** to perform operations as that user. This user's permissions and licensing are in effect when the call to this function returns. The original user must be an administrator. This function can be helpful if you want to debug a problem and see the behavior that the user is experiencing.

When the `ARSetImpersonatedUser` call is issued, the new user's permissions and licensing take effect. When the server receives an API call with this property, the server validates that the user who is logged in is actually an administrator. The server then skips password validation and creates a user record for the impersonated user. If the call requires licenses, the server acquires the needed licenses on behalf of the user.

When the name of the impersonated user is set to `NULL`, the API automatically calls the `ReleaseCurrentUser` function for the impersonated user. The server switches back to the original administrator user.

Note

You can issue the `ARSetImpersonatedUser` call to impersonate a different user instead of setting the name to `NULL`. The permissions and licensing of the new user take effect, and the previously impersonated user is released.

For more information, see ARSetImpersonatedUser.

You can also impersonate a user through the driver program with the `imuser` command. For more information about the driver program, see Using the driver program.

7.9 Enabling client-managed transactions

A client can perform multiple API calls within a single transaction to ensure that all operations are committed or rolled back together, and to ensure that later API calls within the transaction can see operations that previous calls in the transaction performed in the database.

Note

Bulk API calls perform multiple API calls that write to the database in a single transaction and are more efficient than using client-managed transactions. Use these calls unless you must perform queries within a transaction that can return data from previous operations in the same transaction.

To allow clients to begin and end database transactions, BMC Remedy AR System includes the following functions:

- ARBeginClientManagedTransaction
- AREndClientManagedTransaction

All entry APIs called between these two functions take place in the *same* database transaction. The calls are:

- `GetEntry`
- `CreateEntry`
- `GetEntryBlob`
- `GetMultipleEntries`
- `SetEntry`
- `DeleteEntry`
- `MergeEntry`
- `GetListEntry`
- `GetEntryStatistics`
- `ServiceEntry`
- `GetListEntryWithFields`
- `XMLCreateEntry`
- `XMLSetEntry`
- `XMLGetEntry`
- `XMLServiceEntry`
- `XMLDeleteEntry`
- `BulkEntry`
- `BulkEntryCommit`
- `BulkEntryRollback`
- `BulkEntryBegin`
- `GetListEntryBlocks`
- `GetOneEntryWithFields`

- `GetListEntryWithMultiSchemaFields`

A client can issue CMDB *and* BMC Remedy AR System API calls that take place in the same database transaction. To allow this, BMC Remedy AR System also includes the following functions:

- ARSetClientManagedTransaction
- ARRemoveClientManagedTransaction

Unique handles are generated for every new transaction that is started so that clients cannot use old handles to access other client-managed transactions. Any call that includes a handle that the server cannot find returns an error.

At the end of any these functions, the status is examined. If an error is returned, the entire transaction is rolled back, and the database connection dedicated to the transaction is freed. The handle is removed.

These functions give you control over entry-related APIs (such as `ARMergeEntry`). You cannot use a design-time function (such as `ARCreateSchema`) to get control over the entry-related APIs.

7.10 Sending alerts to a filter API plug-in

You can send alerts to a filter-API plug-in.

7.10.1 To write a plug-in to send alerts

1. Write a filter API plug-in.
   Make sure that the AR System server sends the following parameters to the filter API plug-in:
   - `User`
   - `Entry ID`
   - `Alert Text`
   - `Plug-in values`
   - `Encrypted plug-in values`
     The plug-in must handle data coming through plug-in values and encrypted plug-in values.
2. Configure the **ar.cfg** (**ar.conf**) file to indicate the plug-in server where this plug-in will be loaded.
3. Configure the **pluginsvr_config.xml** file to load the plug-in.
4. Add an entry into the AR System Alert Delivery Registration form for this new plug-in. (The entry here defines a user-friendly name for the plugin.)
5. When adding an entry into the AR System Alert User Registration form for a user, select this plug-in as the **Plugin Name**.
   When the Notify (Alert) filter is executed for this user, the plug-in receives the data.

8 Debugging API programs

Use the following information to understand how to solve problems with your API program by using logging and the **driver** program.

8.1 Logging BMC Remedy AR System activity

Logging the actions of your API program can help you determine why your program is not performing the operation that you expect. For example, you might discover that a filter or escalation is interfering with your program, thereby causing unexpected results.

To turn on and manage most types of log files, use the Log Files tab of the AR System Administration Console: Server Information form. For example, turning on API logging causes the server to record all API calls it receives in the log file that you specify. Each entry in a log file identifies the function name, key parameter values, and the return value. In addition, you can use the ARAPILOGGING environment variable to capture commands to the server and server responses for any application that uses the BMC Remedy AR System C API.

For information about the settings in the Log Files tab of the Server Information form, see Setting log files options. For information about how to use log files and the actions tracked by each type of log file, see Working with logs.

8.2 Using the driver program

The **driver** program provides a command line interface for calling every API function, and viewing return values. On a Windows installation, the **driver** program is installed in the **Api\driver** directory under the server directory. For Linux and UNIX installations, it is installed in the **api/src/driver** directory under the server directory.

It also includes print routines for every data structure in the API, making it a useful manual debugging tool. The **driver** program is not interactive, but you can use it to log in to the BMC Remedy AR System server and observe the results and behavior of the equivalent function calls that you make in your API programs.

After compiling the source code or locating the prebuilt program supplied with the API, you can use **driver** for a number of purposes:

- To identify function input parameters and load them with appropriate values
- To examine the content and structure of function output parameters
- To experiment with different parameter values

8.2.1 To use the driver program

1. Make sure your BMC Remedy AR System server is running.
2. For Windows, double-click the **driver.exe** icon. For UNIX, change directories to the ***ARInstallDir* /api/src/driver** directory and type `driver` at the prompt.
3. Specify the login parameters with the `log` command.
   Use an Administrator login so that you have administrative privileges. Demo is the default system administrator.
4. Initialize an API session with the `init` command.
5. Type the abbreviation of the function call at the command line, and supply the appropriate input parameter values. (For a list of abbreviations, type: ?)

When you are working with the specific commands, see BMC Remedy AR System C API functions to enter the appropriate values for the function parameters. Also, if you are working with specific entries, use leading zeros to see the entry ID of those entries.

Note

The **driver** program is provided as sample code and is not a supported BMC Remedy AR System utility.

Use the following information to understand how to use print.C routines, use the driver program from the command line, and create and use driver scripts:

- Using print.C routines
- Using the driver program from the command line
- Creating and using driver scripts

8.2.2 Using print.C routines

One of the most helpful components of the **driver** program is the set of print routines located in the **print.C** file. These routines enable you to print the contents of any data structure in the API. The routines provide code examples for accessing the various structure members. Printing the contents of a structure before and after an API call is a useful debugging tool.

Note

See the function definitions in **print.C** to determine the specific parameters and their types for these routines.

The following examples illustrate the use of these routines.

Code fragment that calls three print.C functions

```
main()
{
    ...
    PrintARFieldValueList(&fieldList);
    PrintReal("header", &value);
    PrintAREntryIdList("header", "header2", &value);
    ...
}
```

Code fragment that shows the output after BMC Remedy AR System executes the command print_entry fred EX000003 (where the server name is fred and the entry ID is EX000003) numItems = 9 ID=FIELD VALUE ---------------------------- 1 = (char) EX000003 2 = (char) snoopy 3 = (timestamp) 11/22/94 4 = (char) snoopy 5 = (char) Demo 6 = (timestamp) 11/23/94 7 = (selection) 2 8 = (char) TESTING 15(S.H.) 5 stat hist items 0 - 1/18/95Demo 1 - 1/19/95Demo 2 - 3 - 4 - 1/26/95Demo

The **print.h** file contains a complete list of these routines. Some of the more commonly used routines are:

- `PrintAREntryIdList`
- `PrintAREntryListFieldList`
- `PrintAREntryListFieldStruct`
- `PrintAREntryListList`
- `PrintAREntryListStruct`
- `PrintARFieldValueList`
- `PrintARInternalIdList`
- `PrintARNameList`
- `PrintARPermissionList`
- `PrintARQualifierStruct`
- `PrintARStatusHistoryList`
- `PrintARStatusList`
- `PrintARStatusStruct`

8.2.3 Using the driver program from the command line

After compiling the source code or locating the prebuilt program supplied with the API, you are ready to use the **driver** program . When you execute the program, the system displays the following list of `driver` commands:

```
Active Link    Escalation      Filter        Entry          Entry
 -----------   ----------      ------        -----          -----
 get    (gal)  get    (ges)    get    (gf)   get      (ge)   stats     (stat)
 set    (sal)  set    (ses)    set    (sf)   set      (se)   get BLOB  (geb)
 create (cal)  create (ces)    create (cf)   create   (ce)   getmult   (gme)
 delete (dal)  delete (des)    delete (df)   merge    {me)   getlistblk(gleb)
 getlist(glal) getlist(gles)   getlist(glf)  delete   (de)   getlistw/f(glewf)
 getmult(gmal) getmult(gmes)   getmult(gmf)  getlist  (gle)  getonew/f (goewf)
               run    (resc)                 service  (sve)  gle w/msf (glmsf)
                                             setget   (sge)

 Char Menu     Schema          License       VUI             Encode/Decode
 ---------     ------          -------       ---             -------------
 get    (gc)   get    (gs)     val    (vl)   get     (gv)    enc query(ecqal)
 set    (sc)   set    (ss)     valmult(vml)  set     (sv)    dec query(dcqal)
 create (cc)   create (cs)     create (cl)   create  (cv)    enc assig(ecasn)
 delete (dc)   delete (ds)     delete (dl)   delete  (dv)    dec assig(dcasn)
 getlist(glc)  getlist(gls)    getlist(gll)  getlist (glsv)  enc hstry(echst)
 getmult(gmc)  gls w/a(glsa)   import (iml)  getmult (gmv)   enc diary(ecdia)
 expand (ec)   getmult(gms)    export (exl)                  enc date (ecdat)
 exp ssm(essm) getlist ext(glxsc)                            dec date (dcdat)

 Container     Alert           Info          Control/Logging Thread/Timer
 ---------     -----           ----          --------------- ------------
 get    (gco)  create   (cae)  get svr (gsi) record   (rec)  launch      (lt)
 set    (sco)  register (rfa)  set svr (ssi) stop rec (srec) launch wait (lwt)
 create (cco)  deregistr(dfa)  get stat(gss) open out (oout) release wait(rwt)
 delete (dco)  gla user (glau) srvr chr(gscs)close out(cout) sleep       (st)
 getlist(glco) get count(gac)  clnt chr(gccs)execute  (ex)   random sleep(rst)
 getmult(gmco) decode   (dcam)               bgn loop (bl)   msec sleep  (msst)
                                             end loop (el)

 Schema Field   Init/Term      Misc Lists    Misc            Misc
 ------------  ------------    ----------    ----            ----
 get     (gsf)  init    (init) server(svr)   ver user (ver)  get file   (gfl)
 set     (ssf)  term    (term) group (glg)   export   (exp/f)set file   (sfl)
 create  (csf)  help    (h, ?) user  (glu)   import   (imp)  get errmsg (gem)
 delete  (dsf)  exit    (e, q) sql   (glsql) unimport (unimp)set logging(slog)
 getlist (glsf) login   (log)  sql al(sqlal) exec proc(proc) close conn (cnc)
 getmult (gmsf) dr ver  (drver)role  (glr)   exec p al(epal) valid cache(vfc)
 setmult (smsf) getconf (gsc)                load qual(lqs)  signal     (sig)
 crtmult (cmsf) setconf (ssc)                set port (ssp)  getmult ep (gmep)
 delmult (dmsf)                              obj chgs (goct) imper user (imusr)
 getmult ext cands(gmxfc)                    get cache event (gce)
 set w/file image (ssfwi)                    create overlay (co)
 create w/file image (csfwi)                 create overlay from obj (cofo)

XML     Transaction             Image        Workflow Debug Commands
---     -----------             -----        ----------------------
get    (xmlge) bgnclnt (bcmt) get     (gi)   cur loc  (dbcl) get KW   (dbkw)
set    (xmlse) endclnt (ecmt) set     (si)   execute  (dbex) user ctxt(dbuc)
create (xmlce) setclnt (scmt) create  (ci)   get FVL  (dbgf) set bkp  (dbbp)
service(xmlsv) relclnt (rcmt) delete  (di)   set FVL  (dbsf) bp clr   (dbbc)
parse  (xmlpd) bgnentry(bbet) getlist (gli)  get qual (dbgq) bp lst   (dbbl)
               endentry(ebet) getmult (gmi)  set qual (dbsq) clr all  (dbbx)
                                             get mode (dbgm) term API (dbta)
                                             set mode (dbsm)


 Localized Val  Currency Ratio   App States
```

```
------------   -------------    ----------
get    (glv)   get    (gcr)     get    (gas)
getmult(gmlv)  getmult(gmcrs)   set    (sas)
                                getlist(glas)
```

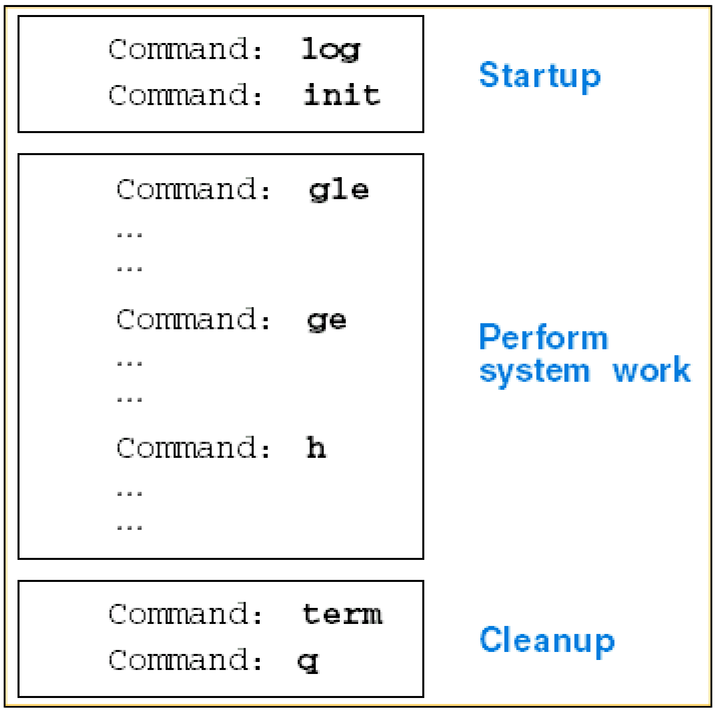
Note

Use the `help` command (h or ?) to display the `driver` commands.

As in all API programs, you must provide the necessary login information and perform initialization operations for connecting to the BMC Remedy AR System server. You can then use the commands to call any number of API functions.

When you are finished, you must terminate your interaction with the BMC Remedy AR System server and exit the **driver** program. The following figure illustrates these high-level steps by using sample `driver` commands.

Order of driver commands



Unlike calling the API functions directly, the `driver` interface prompts you for each input parameter. You can also specify **driver** logon information by using one of the following methods:

- Use the following command-line options:

```
-u   user
-p   password
-l   language
-s   server
```

For example:

```
driver -u Demo -p "fun4me" -s fred
```

- Use the `log` command before or after using the `init` command.

8.2.4 Creating and using driver scripts

This section describes how to write and execute a driver script.

To create a driver script

1. Start the `driver` program.
2. Enter `rec`, and provide *scriptFile* when prompted.
3. Enter `log`, and provide login information when prompted.
4. Enter `init` (initialize BMC Remedy AR System connection).
5. Enter the desired `driver` commands.
6. Enter `term` (terminate BMC Remedy AR System connection).
7. Enter `srec` (stop recording).

Because `driver` scripts are stored as text files, you can edit them by using a text editor (which is sometimes easier than rerecording). Blank lines indicate default or skipped values and should not be deleted.

You have the following options for executing a `driver` script:

Interactive	• Start the `driver` program. • Enter `ex`, and provide *scriptFile* when prompted.
Command Line	• Enter `driver < script.filename`. The driver program terminates at the end of the script.
Command Line or Shortcut	• Enter `driver -x script.filename`. The driver program keeps running at the end of the script unless the script terminates it with a quit (`q`) command. To create a shortcut or application icon that executes your script from a double-click, use this syntax for the shortcut's Target or Command property. It also works directly from the command line.

9 BMC Remedy AR System plug-in API functions

Use the following information to learn about the C BMC Remedy AR System plug-in application programming interface (API) functions and data structures.

- AR System plug-in API functions
- AREA plug-in API functions
- ARDBC plug-in API functions
- AR Filter API plug-in functions

For detailed information about plug-ins, see Enabling Plug-ins. For information about the Java plug-in API, see BMC Remedy AR System plug-in API functions.

9.1 AR System plug-in API functions

Implement these API functions in every BMC Remedy AR System plug-in:

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

9.1.1 Plug-in API data structures

The plug-in API uses the same data structures that the C API uses.

9.1.2 ARPluginCreateInstance

Description

This optional function creates a plug-in instance and can also create instance-specific data. Simple plug-ins might not need to define this function because they do not need to create instance-specific data.

The plug-in server uses multiple threads for improved scalability or throughput. A plug-in can use this function and its context object argument to make instance-specific data thread-safe.

Synopsis

```
#include "arplugin.h"

int ARPluginCreateInstance(
   void **object,
   ARStatusList *status)
```

Return arguments

object

Pointer to the plug-in instance that this function creates, if any. The other functions have a parameter that points to this object to establish the context for the function, so, to be thread-safe, this object must contain all instance-specific data.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARPluginDeleteInstance, ARPluginIdentify, ARPluginInitialization, ARPluginTermination.

9.1.3 ARPluginDeleteInstance

Description

When a thread encounters an exception, the plug-in server calls this optional function to free resources associated with the instance. Define this function only if your plug-in needs to free resources associated with this instance.

Synopsis

```
#include "arplugin.h"

int ARPluginDeleteInstance(
   void *object,
   ARStatusList *status)
```

Return arguments

object

Pointer to the plug-in instance to delete. This is the object that the server returns when it executes the `ARPluginCreateInstance` call.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARPluginCreateInstance, ARPluginIdentify, ARPluginInitialization, ARPluginTermination.

9.1.4 ARPluginEvent

Description

A call to this optional function notifies the plug-in that events have occurred. The plug-in can use this to react to configuration changes.

To be thread-safe, your plug-in must protect any global information or resources accessed here with appropriate mutual exclusion locks.

Synopsis

```
#include "area.h"

int ARPluginEvent(
   ARPropList eventID,
   ARStatusList *status)
```

Input arguments

eventID

Type of event that has occurred:

AR_PLUGIN_EVENT_CONFIG	The BMC Remedy AR System server has informed the plug-in server of a change to the server configuration file.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

9.1.5 ARPluginIdentify

Description

This function provides the type, name, and version of a plug-in. The plug-in server uses this information to identify the plug-in. The server calls this function when it loads a plug-in.

Important

If you do not define this function, the plug-in server does not load the plug-in.

Synopsis

```
#include "arplugin.h"

int ARPluginIdentify(
   ARPluginIdentification *id,
   ARStatusList *status)
```

Return arguments

id

Plug-in type, name, and version.

type

Type of plug-in in this form: **AR_PLUGIN_TYPE_*Type***

name

Unique name of the plug-in.

version

Plug-in version, which is appended to the plug-in type: **AR_PLUGIN_*Type*_VERSION**

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARPluginCreateInstance, ARPluginDeleteInstance, ARPluginInitialization, ARPluginTermination.

9.1.6 ARPluginInitialization

Description

The plug-in server calls this optional function when the it loads the plug-in so the plug-in can allocate and initialize global resources. The plug-in can use these resources for the life of the plug-in instance.

Synopsis

```
#include "arplugin.h"

int ARPluginInitialization(
   ARStatusList *status)
```

Return arguments

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARPluginCreateInstance, ARPluginDeleteInstance, ARPluginIdentify, ARPluginTermination.

9.1.7 ARPluginSetProperties

Description

The plug-in server calls this optional function immediately after `ARPluginIdentify` to provide information to the plug-in. The information can include plug-in service configuration parameters or services that the plug-in server provides to the plug-ins it loads.

To be thread-safe, your plug-in must protect any global information or resources accessed here with appropriate mutual exclusion locks.

Synopsis

```
#include "area.h"

int ARPlugInSetProperties(
   ARPropList *propList,
   ARStatusList status)
```

Input arguments

propList

List of `ARPropStruct` structures. The following property is supported:

`AR_PLUGIN_PROP_LOG_FUNCTION`	An ARByteList containing a function pointer to the plug-in logging facility. The plug-in can use this function pointer to log

	plug-in-specific information to the plug-in log file. For information about plug-in logging, see Server logging modes and Setting log files options.

Return values

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

Note

To avoid confusion if multiple plug-ins choose the same error code, include some identification in the associated text loaded into the status array of the externally written plug-in. For example, prefix the error message with "Employee Plug-in:" or "My Plug-in:".

See also

ARPluginCreateInstance, ARPluginDeleteInstance, ARPluginIdentify, ARPluginTermination, ARPluginInitialization

9.1.8 ARPluginTermination

Description

This optional function deallocates global resources for a plug-in instance when the plug-in instance completes its function.

Synopsis

```
#include "arplugin.h"

int ARPluginTermination(
   ARStatusList *status)
```

Return arguments

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARPluginCreateInstance, ARPluginDeleteInstance, ARPluginIdentify, ARPluginInitialization.

9.2 AREA plug-in API functions

Implement these API functions in BMC Remedy AR System External Authentication (AREA) plug-ins:

- AREAFreeCallback
- AREANeedToSyncCallback
- AREAVerifyLoginCallback

9.2.1 AREA API data structure

The AREA API data structure returns user login information from the AREA plug-in to the BMC Remedy AR System server through **AREAResponseStruct** structure. **AREAResponseStruct** contains information about login status (whether the login attempt succeeded or failed or the user is unknown), license information for modifying entries and performing full text search (FTS), and message information that can be displayed to a user or logged to a file (**aruser.log**).

If you configure the system to cross-reference a blank password, the **AREAResponseStruct** element that the AREA plug-in returns overrides the related value in the user entry on the User form. For example, if the plug-in returns a value for the **groups** element of **AREAResponseStruct**, those groups override the groups in the user entry on the User form.

The AREA plug-in cannot return a fixed license (**AR_LICENSE_TYPE_FIXED**). Users must have a fixed license selected in their entry in the User form.

The **AREAResponseStruct** structure is defined in the **area.h** file.

An AREA plug-in implementation must use the data structure to return a response. The structure contains the following elements:

licenseMask	An integer that indicates the type of licenses for the user. To preserve license values in the User form, set **licenseMask** to zero (0). AREA_LICENSE_MASK_WRITE: Setting for write license AREA_LICENSE_MASK_FULL_TEXT: Setting for FTS license AREA_LICENSE_MASK_RESERVED1: Reserved AREA_LICENSE_MASK_ALL: All of these options (for convenience)
licenseWrite	An integer that indicates the type of license for the user. AR_LICENSE_TYPE_NONE: Indicates no write license (read license) AR_LICENSE_TYPE_FLOATING: Indicates floating write license
licenseFTS	

	An integer that indicates the FTS license type for the user. AR_LICENSE_TYPE_NONE: Indicates no FTS license AR_LICENSE_TYPE_FLOATING: Indicates floating FTS license
licenseRes1	An integer that indicates the type of licenses for the user. This is reserved.
licenseApps	A character string of the application licenses, separated by semicolons, to which the user is licensed.
groups	A character string of the group names, separated by semicolons, to which the user belongs. To preserve the group information in the User form, set groups to NULL.
notifyMech	An integer that indicates the type of notification for the user. This can be notification by email or through BMC Remedy Alert.* To preserve the notify mechanism in the User form, set notifyMech to zero (0). AR_NOTIFY_VIA_NONE: No notifications is sent AR_NOTIFY_VIA_NOTIFIER: Notifications sent through BMC Remedy Alert AR_NOTIFY_VIA_EMAIL: Notifications sent through email AR_NOTIFY_VIA_DEFAULT: Notifications sent according to the setting in the User form
email	The email address for the user. To preserve the email address in the User form, set email to NULL.
loginStatus	Indication that the login succeeded or failed or that the user is unknown. AREA_LOGIN_SUCCESS: Successful login (authenticated) AREA_LOGIN_UNKNOWN_USER: User is not known to the system AREA_LOGIN_FAILED: Failed to authenticate login
messageText	The message text that might appear to users. For example: "You failed to log in." or "Your password will expire in less than 30 days." If the login attempt fails, this appears as an AR_RETURN_ERROR. On successful login, this appears as AR_RETURN_OK , and the message appears as a note, not an error.
logText	Messages that only an administrator uses; these appear in the aruser.log file.
modTime	The last time that the user's information changed. This is represented in the number of seconds from January 1, 1970 (UNIX time). If modTime is 0, the timestamp parameter has not changed since the last time a time stamp was specified.

* BMC Remedy Alert is no longer shipped with BMC Remedy AR System.

The **licenseMask** information is necessary because the license information normally contained in the User form is overridden when you use external authentication. This is relevant in situations where some of your BMC Remedy AR System users are listed in the User form and some are authenticated externally.

This is the AREA response data structure:

```
typedef struct AREAResponseStruct {
   unsigned int licenseMask; /* AREA_LICENSE_MASK_* */
   unsigned int licenseWrite;/* AR_LICENSE_TYPE_* from AR API */
   unsigned int licenseFTS; /* AR_LICENSE_TYPE_* from AR API */
   unsigned int licenseRes1; /* AR_LICENSE_TYPE_* from AR API */
   char         *licenseApps; /* AR_LICENSE_TYPE_* from AR API */
   char         *groups; /* semi-colon separated list*/
   unsigned int notifyMech; /* AR_NOTIFY_* from AR API */
   char         *email;
unsigned int loginStatus; /* AREA_LOGIN_* */
   char         *messageText; /* text seen by user */
   char         *logText; /* text placed in user log */
   ARTimestamp     *modTime;
   AREAResponseStruct;
```

9.2.2 AREAFreeCallback

Description

This optional function frees the data associated with the `AREAResponseStruct`, which the `AREAVerifyLoginCallback` function returns. Although the plug-in 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 server calls this function so the plug-in can deallocate the storage. This allows the plug-in to cache information, if needed.

The plug-in server calls this function after it calls the `AREAVerifyLoginCallback` call. The `AREAVerifyLoginCallback` call returns a pointer to the response structure as a parameter.

To be thread-safe, your plug-in must protect any global information or resources accessed here with appropriate mutual exclusion locks.

An AREA plug-in implementation must return a response. A `NULL` response indicates a failed login attempt.

Important

You must free whatever storage is allocated for the purpose of this response only. For example, if you get the response data from a prebuilt cache constructed during initialization, you do not free anything here. If you do not get the data from a prebuilt cache, you should deallocate response data here.

Synopsis

```
#include "area.h"

int AREAFreeCallback(
   void *object,
   AREAResponseStruct *response)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

response

Pointer to the response structure that the call to `AREAVerifyLoginCallback` returns.

See also

AREAVerifyLoginCallback.

9.2.3 AREANeedToSyncCallback

Description

This function determines whether the user information in the server cache is invalid. The BMC Remedy AR System server periodically checks with the AREA plug-in to determine whether any in-memory copies of user information have expired. The plug-in server calls this function if it receives an authentication request from the server at least five minutes since the last authentication request.

To be thread-safe, your plug-in must protect any global information or resources accessed here with appropriate mutual exclusion locks.

A nonzero return value instructs the BMC Remedy AR System server to expire its internally stored user information.

Synopsis

```
#include "area.h"

int AREANeedToSyncCallback(
   void *object)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returns.

9.2.4 AREAVerifyLoginCallback

Description

The plug-in server calls this function when the BMC Remedy AR System server makes a request to authenticate a user. The BMC Remedy AR System server passes the unencrypted user name and password as parameters.

To be thread-safe, your plug-in must protect any global information or resources accessed here with appropriate mutual exclusion locks.

An AREA plug-in implementation must return a response. A NULL response indicates a failed login attempt.

The BMC Remedy AR System server detects the IP address of the client and sends it to the plug-in, which rejects or accepts the requests.

Synopsis

```
#include "area.h"

int AREAVerifyLoginCallback(
   void *object,
   ARAccessNameType user,
   ARPasswordType password,
   ARAccessNameType networkAddr,
   ARAuthType authString,
   AREAResponseStruct **response)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

user

User name.

password

User password. If the password is a NULL pointer, as opposed to an empty string, the BMC Remedy AR System is attempting to send the user a notification and needs to determine the user's notification mechanism and email address. When the password is NULL, the BMC Remedy AR System is not requesting authentication of the user's identity. The plug-in implementation should return notification mechanism and email address information for the user and return a successful login attempt.

networkAddr

IP address of the BMC Remedy AR System client. The BMC Remedy AR System server determines the client's IP address and passes the address, user name, and password to the plug-in.

For connections using BMC Remedy AR System external authentication (AREA), you can use the IP address of the client machine as authentication, even when using a firewall or a load balancer. However, you cannot use the IP address as authentication when making mid tier connections. In this case, the IP address passed is for the machine where the mid tier is installed, not for the client machine.

authString

String that the BMC Remedy AR System API client user enters into the login screen. This string has no format restrictions. The AREA plug-in uses this string for authentication purposes.

Return values

response

Pointer to a response structure containing user information returned from the plug-in. The structure should be freed using `AREAFreeCallback` when no longer needed.

See also

AREAFreeCallback.

9.3 ARDBC plug-in API functions

Implement these API functions in BMC Remedy AR System Database Connectivity (ARDBC) plug-ins:

- ARDBCCommitTransaction
- ARDBCCreateEntry
- ARDBCDeleteEntry
- ARDBCGetEntry
- ARDBCGetEntryBLOB
- ARDBCGetEntryStatistics
- ARDBCGetListEntryWithFields
- ARDBCGetMultipleFields
- ARDBCRollbackTransaction
- ARDBCSetEntry

9.3.1 ARDBC API data structure

The ARDBC API uses the same data structures as the C API in addition to the **ARVendorFieldStruct** structure. The **ARVendorFieldStruct** structure is defined in the **ardbc.h** header file.

9.3.2 ARDBCCommitTransaction

Description

This optional function 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 calls this function, the BMC Remedy AR System server begins the next transaction.

Important

If you do not define this function and the plug-in server receives a *commit transaction* request, the call proceeds without error.

Synopsis

```
#include "ardbc.h"

int ARDBCCommitTransaction(
   void *object,
   ARInternalId transId,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

transId

Transaction identification number (transaction ID) that the plug-in server designates for each transaction. The transaction ID is unique to the thread that the BMC Remedy AR System server uses to access the ARDBC plug-in. There is only one open transaction per server thread.

Return values

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCRollbackTransaction.

9.3.3 ARDBCCreateEntry

Description

This function creates an entry in the external data source. The plug-in server calls this function on request from the BMC Remedy AR System server. The BMC Remedy AR System server send a request when:

- An BMC Remedy AR System API client calls the `ARCreateEntry` or `ARMergeEntry` function.
- Users enter information to create or merge entries into an external form.
- Users import entries into an external form.
- The Distributed Server Option (DSO) transfers or merges entries into an external form.
- A Push Fields filter or an escalation action creates a new entry in an external form.

Important

If you do not define this function, your plug-in does not create an entry and the BMC Remedy AR System server receives an error message from the plug-in server.

Synopsis

```
#include "ardbc.h"

int ARDBCCreateEntry(
   void *object,
   char *tableName,
   ARVendorFieldList *vendorFieldList,
   ARInternalId transId,
   ARFieldValueList *fieldValueList,
   AREntryIdList *entryId,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

tableName

Name of the external table on which the plug-in creates an entry.

vendorFieldList

List of external form fields. The plug-in uses this list to map BMC Remedy AR System field identification numbers (field IDs) to corresponding external form field names. All other input arguments and return values must use the BMC Remedy AR System field IDs.

transId

Transaction identification number (transaction ID) that the plug-in server designates for each transaction. The transaction ID is unique to the thread that the BMC Remedy AR System server uses to access the ARDBC plug-in. There is only one open transaction per server thread.

fieldValueList

List of the data in a new entry. The list consists of one or more field and value pairs that the server specifies in any order.

Return values

entryId

Unique identifier for the entry that the function creates. It must correspond to a value in the external data source and must be unique, non-NULL, and either character or numeric data. For an external data source, the entry ID can have more than 15 characters. Therefore, the entry ID can consist of one or more values of type **AREntryIdType** and is represented by the `AREntryIdList` structure.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCDeleteEntry, ARDBCGetEntry, ARDBCGetEntryBLOB, ARDBCSetEntry.

9.3.4 ARDBCDeleteEntry

Description

This function deletes a single entry in an external data source. The plug-in server calls this function when it receives a delete entry request from the BMC Remedy AR System server. The BMC Remedy AR System server send a request when:

- An BMC Remedy AR System API program calls the `ARDeleteEntry` function.
- Users delete entries in an external form.
- The server receives a delete entry command from a workflow process to delete an entry.

Important

If you do not define this function, the BMC Remedy AR System server receives an error message and does not delete the entry.

Synopsis

```
#include "ardbc.h"

int ARDBCDeleteEntry(
   void *object,
   char *tableName,
   ARVendorFieldList *vendorFieldList,
   ARInternalId transId,
   AREntryIdList *entryId,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

tableName

Name of the external table on which the plug-in deletes this entry.

vendorFieldList

List of external form fields. The plug-in uses this list to map BMC Remedy AR System field identification numbers (field IDs) to corresponding external form field names. All other input arguments and return values must see the BMC Remedy AR System field IDs.

transId

Transaction identification number (transaction ID) that the plug-in server designates for each transaction. The transaction ID is unique to the thread that the BMC Remedy AR System server uses to access the ARDBC plug-in. There is only one open transaction per server thread.

entryId

Unique identifier for the entry that to delete. It must correspond to a value in the external data source and must be unique, non-NULL, and either character or numeric data. For an external data source, the entry ID can have more than 15 characters. Therefore, the entry ID can consist of one or more values of type **AREntryIdType** and is represented by the `AREntryIdList` structure.

Return values

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCCreateEntry, ARDBCGetEntry, ARDBCGetEntryBLOB, ARDBCSetEntry.

9.3.5 ARDBCGetEntry

Description

This function retrieves a single entry from an external data source table. The plug-in server calls this function when the BMC Remedy AR System server receives an `ARGetEntry` request from a client.

Synopsis

```
#include "ardbc.h"

int ARDBCGetEntry(
   void *object,
   char *tableName,
   ARVendorFieldList *vendorFieldList,
   ARInternalId transId,
   AREntryIdList *entryId,
   ARInternalIdList *idList,
   ARFieldValueList *fieldList,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

tableName

Name of the external table from which the plug-in retrieves this entry.

vendorFieldList

List of external form fields. The plug-in uses this list to map BMC Remedy AR System field identification numbers (field IDs) to corresponding external form field names. All other input arguments and return values must see the BMC Remedy AR System field IDs.

transId

Transaction identification number (transaction ID) that the plug-in server designates for each transaction. The transaction ID is unique to the thread that the BMC Remedy AR System server uses to access the ARDBC plug-in. There is only one open transaction per server thread.

entryId

Unique identifier for the entry to retrieve. It must correspond to a value in the external data source and must be unique, non-NULL, and either character or numeric data. For an external data source, the entry ID can have more than 15 characters. Therefore, the entry ID can consist of one or more values of type **AREntryIdType** and is represented by the `AREntryIdList` structure.

idList

List of zero or more BMC Remedy AR System field identification numbers (field IDs) for the fields to retrieve. If the **idList** value is `NULL` or the list contains zero field IDs, the function retrieves all the fields.

Return values

fieldList

List of the data in the entry that this function retrieves. The list consists of one or more field and value pairs that the BMC Remedy AR System server specifies in any order.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCCreateEntry, ARDBCDeleteEntry, ARDBCGetEntryBLOB, ARDBCSetEntry.

9.3.6 ARDBCGetEntryBLOB

Description

This function opens an attachment. When an BMC Remedy AR System client calls the `ARGetEntryBLOB` API function to open an attachment from an external form, the BMC Remedy AR System server sends a request and the plug-in server calls the `ARDBCGetEntryBLOB` function.

Important

If you do not define this function and the plug-in server receives a *get entry BLOB* request, the BMC Remedy AR System server receives an error message and does not retrieve the data.

Synopsis

```
#include "ardbc.h"

int ARDBCGetEntryBLOB(
   void *object,
   char *tableName,
   ARVendorFieldList *vendorFieldList,
   ARInternalId transId,
   AREntryIdList *entryId,
   ARInternalId fieldId
   ARLocStruct *loc
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

tableName

Name of the external table that contains the attachment to open.

vendorFieldList

List of external form fields. The plug-in uses this list to map BMC Remedy AR System field identification numbers (field IDs) to corresponding external form field names. All other input arguments and return values must see the BMC Remedy AR System field IDs.

transId

Transaction identification number (transaction ID) that the plug-in server designates for each transaction. The transaction ID is unique to the thread that the BMC Remedy AR System server uses to access the ARDBC plug-in. There is only one open transaction per server thread.

entryId

The unique identifier for the entry that this function retrieves. This must correspond to a value in the external data source and must be unique, non-NULL, and either character or numeric data. For an external data source, the entry ID can have more than 15 characters. Therefore, the entry ID can consist of one or more values of type **AREntryIdType** and is represented by the `AREntryIdList` structure.

fieldId

Field ID for the field that this function retrieves.

Return values

loc

Pointer to an `ARLocStruct` structure that contains the name of the buffer.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCCreateEntry, ARDBCDeleteEntry, ARDBCGetEntry, ARDBCSetEntry.

9.3.7 ARDBCGetEntryStatistics

Description

This function gathers statistical information about a set of data. The plug-in server calls this function when it receives a request from the BMC Remedy AR System server to get entry statistics.

Important

If you do not define this function and the plug-in server receives a *get entry statistics* request, the BMC Remedy AR System server receives an error message and does not retrieve the data.

Synopsis

```
#include "ardbc.h"

int ARDBCGetEntryStatistics(
   void *object,
   char *tableName,
   ARVendorFieldList *vendorFieldList,
   ARInternalId transId,
   ARQualifierStruct *qualifier,
   ARFieldValueOrArithStruct *target,
   unsigned int statistic,
   ARInternalIdList *groupByList,
   ARStatisticsResultList *results,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

tableName

Name of the external table from which to retrieve the statistics.

vendorFieldList

List of external form fields. The plug-in uses this list to map BMC Remedy AR System field identification numbers (field IDs) to corresponding external form field names.

Note

All other ARDBC input arguments and return values see BMC Remedy AR System field IDs, not external form IDs.

transId

Transaction identification number (transaction ID) that the plug-in server designates for each transaction. The transaction ID is unique to the thread that the BMC Remedy AR System server uses to access the ARDBC plug-in. There is only one open transaction per server thread.

qualifier

Query that determines the set of entries to retrieve. The qualification can include one or more fields and any combination of conditional, relational, and arithmetic operations (numeric data only).

target

Arithmetic operation that defines the statistic to compute. The statistic can include one or more fields and any combination of arithmetic operations. The system generates an error if the user does not have read permission for a field or if a field does not exist. If you specify `AR_STAT_OP_COUNT` for the `statistic` parameter, assign a tag value of `0` to omit this parameter.

statistic

Value that indicates the statistic type.

1:	The total number of matching entries (`AR_STAT_OP_COUNT`).
2:	The sum of values for each group (`AR_STAT_OP_SUM`).
3:	The average value for each group (`AR_STAT_OP_AVERAGE`).
4:	The minimum value for each group (`AR_STAT_OP_MINIMUM`).
5:	The maximum value for each group (`AR_STAT_OP_MAXIMUM`).

groupByList

List of zero or more fields to group the results by. The system computes a result for each group of entries having the same value in the specified field. Specifying more than one field creates groups within groups, each of which returns a separate statistic. Specify `NULL` for this parameter (or zero fields) to compute a single result for all matching entries.

Return values

results

List of zero or more results. If you specify one or more fields for the `groupByList` parameter, each item in the list represents a group. Each result structure contains the field values that define the group and the statistic for that group.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCGetEntry.

9.3.8 ARDBCGetListEntryWithFields

Description

This function retrieves a list of qualified entries with field data. The plug-in server calls this function when the BMC Remedy AR System server receives an `ARGetListEntryWithFields` or an `ARGetListEntry` request from a client.

⚠ **Important**

If you do not define this function and the plug-in server receives a *get list entry with fields* request, the BMC Remedy AR System server returns a list of zero entries.

Synopsis

```
#include "ardbc.h"

int ARDBCGetListEntryWithFields(
   void *object,
   char *tableName,
   ARVendorFieldList *vendorFieldList,
   ARInternalId transId,
   ARQualifierStruct *qualifier,
   ARSortList *sortList,
   AREntryListFieldList *getListFields,
   unsigned int firstRetrieved,
   unsigned int maxRetrieve,
   AREntryListFieldValueList *entryList,
   unsigned int *numMatches,
   ArStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

tableName

Name of the external table from which the plug-in retrieves the entry.

vendorFieldList

A list of external form fields. The plug-in uses this list to map BMC Remedy AR System field identification numbers (field IDs) to corresponding external form field names. All other input arguments and return values must see the BMC Remedy AR System field IDs.

transId

Transaction identification number (transaction ID) that the plug-in server designates for each transaction. The transaction ID is unique to the thread that the BMC Remedy AR System server uses to accesses the ARDBC plug-in. There is only one open transaction per server thread.

qualifier

Query that determines the set of entries that this function retrieves. Values for this argument might include one or more fields and any combination of conditional, relational, and arithmetic operations (for numeric data types only).

sortList

List of zero or more fields that identifies the order in which the function sorts the entries.

getListFields

List of zero or more fields that the function retrieves with each entry.

firstRetrieved

First entry to retrieve. The function ignores entries that occur before the entry that this argument specifies.

maxRetrieve

Maximum number of entries to retrieve. You can configure this option at both the BMC Remedy AR System client and BMC Remedy AR System server levels. The value that the BMC Remedy AR System server passes when it calls this function is the lower of the two values.

Return values

entryList

List of zero or more retrieved entries.

numMatches

Total number of entries that match the criteria in the `qualifier` argument. This value is the same as the number of entries that the function returns if the number of matching entries is less than or equal to the value in the `maxRetrieve` argument.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCGetListSchemas.

9.3.9 ARDBCGetListSchemas

Description

This function retrieves a list of external schemas. The plug-in server calls this function when the BMC Remedy AR System server receives an `ARGetListExtSchemaCandidates` request from the client to list candidate schemas.

To test your ARDBC plug-in in a vendor form, you must implement `ARDBCGetListSchemas`. If you do not implement this function, the Available Window Names and Available Vendor Tables fields in the Vendor form do not contain the name of your plug-in.

⚠ **Important**

If you do not define this function and the plug-in server receives a *get list schemas* request, the BMC Remedy AR System server returns a list of zero forms.

Synopsis

```
#include "ardbc.h"

int ARDBCGetListSchemas(
   void *object,
   ARCompoundSchemaList *schema,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

Return values

schema

List of retrieved schemas.

The name specified in **schema-compoundSchema[n].u.vendor.vendorName** must match the plug-in name specified in `ARPluginIdentify` in the id-name field.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCGetListEntryWithFields, ARDBCGetMultipleFields.

9.3.10 ARDBCGetMultipleFields

Description

This function retrieves a list of fields from an external data source. The plug-in calls this function when the BMC Remedy AR System server receives an `ARGetMultipleExtFieldCandidates` API function from a client. For example, an BMC Remedy Developer Studio client might provide a user with a list of fields instead of requiring the user to enter a field name.

To test your ARDBC plug-in in a vendor form, you must implement `ARDBCGetMultipleFields`. If you do not implement this function, the Available Column fields in the Vendor form do not contain the name of your plug-in.

Important

If you do not define this function and the plug-in server receives a *get multiple fields* request, the BMC Remedy AR System server returns a list of zero fields.

Synopsis

```
#include "ardbc.h"

int ARDBCGetMultipleFields(
   void *object,
   ARCompoundSchema *schema,
   ARFieldMappingList *mapping,
   ARFieldLimitList *limit,
   ARUnsignedIntList *dataType,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

schema

`ARCompoundSchema` structure that describes all external forms.

Return values

mapping

List of zero or more `ARFieldMapping` structures that describe external form fields.

limit

List of zero or more `ARFieldLimit` structures, each of which corresponds to the fields that the BMC Remedy AR System server returns.

dataType

List of zero or more integers that represent the data types of the retrieved fields. See the **ar.h** file for the integer definitions, such as the `AR_DATA_TYPE_CHAR` definition.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCGetEntry, ARDBCGetListEntryWithFields, ARDBCGetListSchemas.

9.3.11 ARDBCRollbackTransaction

Description

This function rolls back the changes of one or more previous ARDBC calls to the external data source. The plug-in server calls this function when one or more actions that read or modify the external data source result in a failed operation or an error. After the plug-in server issues this function, the BMC Remedy AR System server begins the next transaction.

Important

If you do not define this function and the plug-in server receives a *roll back transaction* request, the BMC Remedy AR System server does not process the request and does not return an error.

Synopsis

```
#include "ardbc.h"

int ARDBCRollbackTransaction(
   void *object,
   ARInternalId transId,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

transId

Transaction identification number (transaction ID) that the plug-in server designates for each transaction. The transaction ID is unique to the thread that the BMC Remedy AR System server uses to access the ARDBC plug-in. There is only one open transaction per server thread.

Return values

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCCommitTransaction.

9.3.12 ARDBCSetEntry

Description

This function modifies the contents of a single entry or BLOB. The plug-in server calls this function when:

- The BMC Remedy AR System server receives a call to `ARSetEntry` or `ARMergeEntry` from the client.
- Users enter information to create or merge entries into an external form.
- A Push Fields filter or an escalation action modifies an entry.

If the specified entry does not exist, the plug-in creates it. An `ARDBCSetEntry` operation on a nonexistent entry can occur when the BMC Remedy AR System server receives a call to `ARMergeEntry` from a client.

Important

If you do not define this function, the BMC Remedy AR System server receives an error message and the entry is not modified.

Synopsis

```
#include "ardbc.h"

int ARDBCSetEntry(
   void *object,
   char *tableName,
   ARVendorFieldList *vendorFieldList,
   ARInternalId transId,
   AREntryIdList *entryId,
   ARFieldValueList *fieldValueList,
   ARTimestamp getTimestamp,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

tableName

Name of the external table on which the plug-in sets this entry.

vendorFieldList

List of external form fields.

The plug-in uses this list to map BMC Remedy AR System field identification numbers (field IDs) to corresponding external form field names. All other input arguments and return values must see the BMC Remedy AR System field IDs.

transId

Transaction identification number (transaction ID) that the plug-in server designates for each transaction. The transaction ID is unique to the thread that the BMC Remedy AR System server uses to access the ARDBC plug-in. There is only one open transaction per server thread.

entryId

Unique identifier for the entry to set. This must correspond to a value in the external data source and must be unique, non-NULL, and either character or numeric data. For an external data source, the entry ID can be longer than 15 characters. Therefore, the entry ID can consist of one or more values of type **AREntryIdType** and is represented by the `AREntryIdList` structure.

fieldValueList

List of the data in a new entry. The list consists of one or more field and value pairs that the server specifies in any order.

getTimestamp

Time stamp that specifies when the client last retrieved the entry. The ARDBC plug-in can provide functionality like that provided by the BMC Remedy AR System server by returning an error if an entry was modified (presumably by another client) after the calling client last retrieved it. For example, the plug-in can store a modification time stamp and compare this value with it to determine whether the entry changed since the last retrieval. If the value of this parameter is `0`, ignore it.

For a description of the BMC Remedy AR System functionality, see ARSetEntry.

Return values

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARDBCCreateEntry, ARDBCDeleteEntry, ARDBCGetEntry, ARDBCGetEntryBLOB.

9.4 AR Filter API plug-in functions

Use the following information to implement the `ARFilterApiCall` function in BMC Remedy AR System Filter (AR Filter) plug-ins.

- AR Filter API data structure
- ARFilterApiCall

9.4.1 AR Filter API data structure

The AR Filter API uses the same data structures that the BMC Remedy AR System C API uses.

9.4.2 ARFilterApiCall

Description

The plug-in server calls this function when it receives filter API Set Fields action data or input values from the BMC Remedy AR System server during workflow operations.

To be thread-safe, your plug-in must protect any global information or resources accessed here with appropriate mutual exclusion locks. You must implement this function in all filter API plug-ins.

Synopsis

```
#include "arfilterapi.h"

int ARFilterApiCall(
   void *object,
   ARValueList *inValues,
   ARValueList *outValues,
   ARStatusList *status)
```

Input arguments

object

Pointer to the plug-in instance that the call to `ARPluginCreateInstance` returned.

inValues

Array of input values that the filter action specifies. BMC Remedy Developer Studio defines the filter action and what data the BMC Remedy AR System server passes to the AR filter API plug-in.

Return arguments

outValues

Array of output values that the plug-in creator defines.

status

List of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking. The BMC Remedy AR System server passes the status results to the corresponding filter or escalation.

See also

ARPluginCreateInstance, ARPluginDeleteInstance, ARPluginIdentify, ARPluginInitialization, ARPluginTermination.

10 XML transformation routines

The BMC Remedy AR System API provides routines to transform BMC Remedy AR System objects into XML format and XML back into BMC Remedy AR System objects. This section contains descriptions of the transformation routines and XML Schemas that describe the XML format of the BMC Remedy AR System objects. The following topics are provided:

- Transforming XML and BMC Remedy AR System objects
- Object API functions
- Using the object XML functions
- Object API function descriptions

10.1 Transforming XML and BMC Remedy AR System objects

To transform XML Schema into BMC Remedy AR System objects and vice versa, you use the XML Schema definition files (**.xsd** files) and the XML API functions. The BMC Remedy AR System XML Schema definition files describe the format and rules for defining the objects.

You can also import from and export to XML files using BMC Remedy Developer Studio, and the import and export CLI programs. To make sure that you can import and export your own XML files and objects with BMC Remedy AR System, check the AR XML Schema definition files for format information.

For individual BMC Remedy AR System XML tag descriptions, see the XML definition files.

10.1.1 Schema definition files

XML Schemas are documents that are used to define and validate the content and structure of XML data.

An XML Schema defines and describes an XML instance document using the XML Schema definition language (XSD). An XML instance document conforms to the standards of the XML Schema. XML Schema elements (elements, attributes, types, and groups) are used to define the structure, content, and relationship of the XML data. XML Schemas can also provide default values for attributes, and elements. For more information, see the World Wide Web Consortium (W3C) website: http://www.w3.org/.

The BMC Remedy AR System XML Schemas describe the format of the XML that is produced and consumed by BMC Remedy AR System. Applications can be built to integrate with the BMC Remedy AR System using XML that conforms to the standards of the BMC Remedy AR System XML Schemas.

The XML schemas can also be used to validate the XML instance documents. This verifies that all of the elements of data exist, are in the expected sequence, and are all of the correct data type. This helps make sure that BMC Remedy AR System can interpret the XML instance document that is received.

10.2 Object API functions

The object API functions are divided into two categories:

- **ARGet** functions, which transform XML objects into BMC Remedy AR System structures.
- **ARSet** functions, which transform BMC Remedy AR System structures into XML objects.

10.2.1 Object manipulation functions

You can perform **get** and **set** operations for each of the following objects:

Active links

- ARGetActiveLinkFromXML
- ARSetActiveLinkToXML

Character menus

- ARGetMenuFromXML
- ARSetMenuToXML

Containers

- ARGetContainerFromXML
- ARSetContainerToXML

DSO

- ARGetDSOMappingFromXML
- ARSetDSOMappingToXML
- ARGetDSOPoolFromXML
- ARSetDSOPoolToXML

Escalations

- ARGetEscalationFromXML
- ARSetEscalationToXML

Fields

- ARGetFieldFromXML
- ARSetFieldToXML

Filters

- ARGetFilterFromXML
- ARSetFilterToXML

Schemas

- ARGetSchemaFromXML
- ARSetSchemaToXML

VUIs

- ARGetVUIFromXML
- ARSetVUIToXML

10.2.2 Other functions for object manipulation

The XML API also includes the following functions to object manipulation:

- ARParseXMLDocument — creates a parsed XML stream for the **ARGet** XML functions.
- ARGetListXMLObjects — retrieves object names and types from an XML document.
- ARSetXMLDocHeaderToXML and ARSetXMLDocFooterToXML — generates the headers and footers of an XML document.

10.3 Using the object XML functions

You can use the XML functions to transform XML documents to BMC Remedy AR System structures, or BMC Remedy AR System structures to XML documents. You can also use the ARGetListXMLObjects call to get a list of the BMC Remedy AR System structures in an XML document.

Make sure that your program calls the `ARInitialization` call at the beginning of its execution. See BMC Remedy AR System C API functions.

10.3.1 Transforming XML documents into BMC Remedy AR System structures

Use the following BMC Remedy AR System XML API functions and structures to transform XML documents into BMC Remedy AR System structures:

ARXMLInputDoc

Supply this structure with the XML document to parse, in one of the following forms:

- `AR_XML_DOC_CHAR_ST` — null-terminated character string (you must free the memory associated with this action)
- `AR_XML_DOC_FILE_NAME` — file name
- `AR_XML_DOC_URL` — URL
  See the **ar.h** file for definitions of this structure.

ARParseXMLDocument

This function parses the XML document and returns a parsed XML stream and a list of object names and types.
The combination of object name and object type must be unique for each object in the XML document. For example, you cannot have two schemas named **ABC** in a single XML document. However, you can have a schema *and* a filter named **ABC** in one XML document, as these are different object types.

ARGet Object FromXML

These functions retrieve the properties of the object from the parsed XML stream and places the properties into the output parameters. Supply **NULL** for any property that you do not want to retrieve.

FreeARXMLParsedStream

This function frees memory associated with the parsed stream. For more information, see **arfree.h**.

10.3.2 Transforming BMC Remedy AR System structures into XML documents

Use the following BMC Remedy AR System API functions and structures to transform BMC Remedy AR System structures into XML documents:

ARXMLOutputDoc

This structure contains the XML document generated from calls to the **Set** functions in one of the following forms:

- `AR_XML_DOC_CHAR_STR` — null-terminated character string
- `AR_XML_DOC_FILE_NAME` — file name (A new file is created. If the file already exists, the new XML contents are appended to the end of the file.)
- `AR_XML_DOC_FILE_HANDLE` — file handle
  See the **ar.h** file for definitions of this structure.

ARSetHeaderToXML

This function generates the XML declaration and start tag for the root element (such as <*?xml...?*>< *root*>).
The server sets the document encoding in the XML declaration to the encoding of your local machine.

ARSet Structure ToXML

This function transforms BMC Remedy AR System structures into XML.

ARSetFooterToXML

This function generates the XML footer, which is the ending tag for the root element (</root>).

FreeARXMLParsedStream

This function frees memory associated with parsed stream.

If you only have a single structure to transform to XML, you can set the **xmlDocHdrFtrFlag** flag to `TRUE` (T) to automatically generate the XML document header and footer, instead of calling the `ARSetHeaderToXML` and `ARSetFooterToXML` functions. If you transform multiple structures to XML and combine them together, set this flag to `FALSE` (F).

10.3.3 Listing BMC Remedy AR System structures

Use the `ARGetListXMLObjects` call to list the BMC Remedy AR System objects in an XML document and retrieve the following object characteristics:

- The object name.
- The object type, which is in the form `AR_STRUCT_ITEM_XML_` ***objectType***, where ***objectType*** is **filter**, **schema**, or other object types.
- The object subtype. For container objects, the subtype is in the form `ARCON_`***containerType***, where ***containerType*** is a guide, application, packing list, or filter guide. Currently only containers have a valid subtype. The subtype value for non-container objects is `-1`.
  If you are only retrieving object names, use the object subtype to increase the retrieval speed.

  See the **ar.h** file for object values.

10.4 Object API function descriptions

This section describes each XML API function call and its components.

Note

In API function descriptions, when a value is said to be specified, it means the value is found within the synopsis for that particular API function.

- ARGetActiveLinkFromXML
- ARGetContainerFromXML
- ARGetDSOMappingFromXML

- ARGetDSOPoolFromXML
- ARGetEscalationFromXML
- ARGetFieldFromXML
- ARGetFilterFromXML
- ARGetImageFromXML
- ARGetListXMLObjects
- ARGetMenuFromXML
- ARGetSchemaFromXML
- ARGetVUIFromXML
- ARParseXMLDocument
- ARSetActiveLinkToXML
- ARSetContainerToXML
- ARSetDSOMappingToXML
- ARSetDSOPoolToXML
- ARSetEscalationToXML
- ARSetFieldToXML
- ARSetFilterToXML
- ARSetImageToXML
- ARSetMenuToXML
- ARSetSchemaToXML
- ARSetVUIToXML
- ARSetXMLDocFooterToXML
- ARSetXMLDocHeaderToXML
- ARGetSchemaWithExtensionsFromXML
- ARGetFilterWithExtensionsFromXML
- ARGetActiveLinkWithExtensionsFromXML
- ARGetEscalationWithExtensionsFromXML
- ARGetContainerWithExtensionsFromXML
- ARGetVUIWithExtensionsFromXML
- ARGetFieldWithExtensionsFromXML
- ARSetSchemaWithExtendedFieldsToXML
- ARSetVUIWithExtendedFieldsToXML

10.4.1 ARGetActiveLinkFromXML

Description

Retrieves information about an active link from a definition in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetActiveLinkFromXML(
   ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType activeLinkName,
   ARNameType appBlockName,
   unsigned int *executionOrder,
   ARWorkflowConnectStruct *workflowConnect,
   ARInternalIdList *accessList,
   unsigned int *executeOn,
   ARInternalId *controlFieldID,
   ARInternalId *focusFieldID,
   unsigned int *enabled,
   ARQualifierStruct *query,
   ARActiveLinkActionList *ifActionList,
   ARActiveLinkActionList *elseActionList,
   ARSupportFileInfoList *supportFileList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char **helpText,
   char **changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   unsigned int *errorActlinkOptions,
   ARNameType errorActlinkName,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

activeLinkName

The active link name. Each active link name must be unique.

appBlockName

For deployable applications, the application block name of the active link.

Return values

executionOrder

A value between `0` and `1000` (inclusive) that determines the active link execution order. When multiple active links are associated with a form, the value associated with each active link determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnect

The list of form names the active link is linked to. The active link must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

accessList

A list of lists containing zero or more groups who can access the fields retrieved. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify `NULL` for this parameter if you do not want to retrieve the permissions.

executeOn

A bitmask that indicates the form operations that trigger the active link. Specify `NULL` for this parameter if you do not want to retrieve this value.

controlFieldID

The ID of the field that represents the button, toolbar button, or menu item associated with executing the active link. The system returns zero if the **executeMask** does not include the `AR_EXECUTE_ON_BUTTON` condition. Specify `NULL` for this parameter if you do not want to retrieve this value.

focusFieldID

The ID of the field associated with executing the active link by pressing Return or selecting a character menu item. The system returns zero if the **executeMask** does not include the `AR_EXECUTE_ON_RETURN` or `AR_EXECUTE_ON_MENU_CHOICE` conditions. Specify `NULL` for this parameter if you do not want to retrieve this value.

enabled

A flag that specifies whether the active link is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A qualification that determines whether the active link is executed. The system returns zero (`AR_COND_OP_NONE`) if the active link has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

ifActionList

The set of actions performed if the condition defined by the `query` parameter is satisfied. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseActionList

The set of actions performed if the condition defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

supportFileList

The support file that the active link uses.

owner

The active link owner. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastModifiedBy

The user who last modified the active link. Specify `NULL` for this parameter if you do not want to retrieve this value.

modifiedDate

The date that the active link was last modified.

helpText

The help text associated with the active link. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the active link.

objPropList

The object properties of the active link.

arDocVersion

The XML document version.

errorActlinkOptions

Reserved for future use. Set to `NULL`.

errorActlinkName

Reserved for future use. Set to `NULL`.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetActiveLinkToXML.

10.4.2 ARGetContainerFromXML

Description

Retrieves information about a container from a definition in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetContainerFromXML(
   ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType containerName,
   ARNameType appBlockName,
   ARPermissionList *permissionList,
   ARInternalIdList *subAdminGrpList,
   ARContainerOwnerObjList *ownerObjectList,
   char **label,
   char **description,
   unsigned int *containerType,
   ARReferenceList *referenceList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char **helpText,
   char **changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
Input arguments
```

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

containerName

The container name. Each container name must be unique.

appBlockName

For a deployable application, this is the application block name of the container.

Return values

permissionList

A list of zero or more groups who can access this container. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify `NULL` for this parameter if you do not want to retrieve this value.

subAdminGrpList

The list of groups that have access to the container.

ownerObjectList

A list of schemas that own this container. This parameter can be `NULL` if the container exists globally.

label

The label for this container. It can be as many as 255 characters long or `NULL`.

description

The description for this container. It can be as many as 2000 characters long or `NULL`.

containerType

The type for this container — either guide (`ARCON_GUIDE`), application (`ARCON_APP`), or a custom type you have defined.

referenceList

A list of pointers to the objects referenced by this container. References can be to internal BMC Remedy AR System objects (for example, guides reference active links and applications reference forms) or to external objects such as URLs or file names. Specify **NULL** for this parameter if you do not want to associate any objects with this container.

owner

The container owner.

lastModifiedBy

The user who last modified the container.

modifiedDate

The date that the container was last modified.

helpText

The help text associated with the container. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the container.

objPropList

The object properties of the container.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetContainerToXML.

10.4.3 ARGetDSOMappingFromXML

Description

Retrieves information about a DSO mapping from a definition in an XML document. For more information about DSO mapping, see Enabling distributed mappings.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetDSOMappingFromXML(
   ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType mappingName,
   ARNameType appBlockName,
   ARNameType fromSchema,
   ARServerNameType fromServer,
   ARNameType toSchema,
   ARServerNameType toServer,
   unsigned int *enabled,
   unsigned int updateCode,
   unsigned int *transferMode,
   unsigned int *mapType,
   unsigned int *rtnMapType,
   unsigned int *defaultMap,
   unsigned int *duplicateAction,
   unsigned int *patternMatch,
   unsigned int *requiredFields,
   long *retryTime,
   char **mapping,
   char **rtnMapping,
   char **matchQual,
   char **excludeFlds
   char **rtnExcludeFlds
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char **helpText,
   char **changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

mappingName

The name of the schema to be exported. Each schema name must be unique.

appBlockName

For a deployable application, this is the application block name of the DSO mapping.

Return values

fromSchema

The name of the source schema. Each schema name must be unique.

fromServer

The name of the source server.

toSchema

The name of the target schema.

toServer

The name of the target server.

enabled

A flag that specifies whether the DSO mapping is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

updateCode

A value that indicates when the system is to update the DSO mapping after changing the entry.

transferMode

The transfer mode.

mapType

The type of mapping to use on transfer.

rtnMapType

The type of mapping to use on update or return.

defaultMap

A value that indicates the default mapping.

duplicateAction

The action to perform on transfer to existing entry.

patternMatch

A value that indicates that pattern matching is enabled.

requiredFields

A value that indicates if required fields can be ignored.

retryTime

The maximum number of seconds that the server waits to remap the schema, before it cancels the operation.

mapping

A value that indicates custom mapping if the **mapType** value is `custom`.

rtnMapping

A value that indicates custom mapping if the **rtnMapType** value is `custom`.

matchQual

The matching qualification.

excludeFlds

A value that indicates the fields to be excluded from transfer mapping when the mapType value is Like IDs or Like Names. The list of field ids is separated by a semi-colon.

rtnExcludeFlds

A value that indicates the fields to be excluded from update or return mapping when the rtnMapType value is Like IDs or Like Names. The list of field ids is separated by a semi-colon.

owner

The owner of the DSO mapping.

lastModifiedBy

The user who last modified the DSO mapping.

modifiedDate

The date that the DSO mapping was last modified.

helpText

The help text associated with the DSO mapping. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the DSO mapping.

objPropList

The properties of the DSO mapping object.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

10.4.4 ARGetDSOPoolFromXML

Description

Retrieves information about a DSO pool from a definition in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"
```

```
int ARGetDSOPoolFromXML(
ARControlStruct *control,
   ARXMLParsedStream  *parsedStream,
   ARNameType poolName,
   ARNameType appBlockName,
   unsigned int *enabled,
   unsigned int *defaultPool,
   long *threadCount,
   char **connection,
   unsigned int *polling,
   unsigned int *pollingInterval,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char **helpText,
   char **changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

poolName

The name of the DSO pool. Each pool name must be unique.

appBlockName

For a deployable application, this is the application block name of the DSO pool.

Return values

enabled

A flag that specifies whether the DSO pool is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

defaultPool

A value that indicates that this is the default pool.

threadcount

The thread count of the DSO pool.

connection

This parameter is reserved for future use.

polling

Indicates whether this pool polls the pending queue for pending items assigned to it. Pools are signaled, by default.

pollingInterval

The polling interval (if the pool is a polling pool).

owner

The owner of the DSO pool.

lastModifiedBy

The user who last modified the DSO pool.

modifiedDate

The date that the DSO pool was last modified.

helpText

The help text associated with the DSO pool. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the DSO pool.

objPropList

The properties of the DSO pool.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

10.4.5 ARGetEscalationFromXML

Description

Retrieves information about an escalation from a definition in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetEscalationFromXML(
   ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType escalationName,
   ARNameType appBlockName,
   AREscalationTmStruct *escalationTime,
   ARWorkflowConnectStruct *workflowConnect,
   unsigned int *enabled,
   ARQualifierStruct *query,
   ARFilterActionList *ifActionList,
   ARFilterActionList *elseActionList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char **helpText,
   char **changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

escalationName

The name of the escalation. Each escalation name must be unique.

appBlockName

For a deployable application, this is the application block name of the escalation.

Return values

escalationTime

The time specification for evaluating the escalation condition. This parameter can take one of two forms: a time interval that defines how frequently the server checks the escalation condition (in seconds) or a bitmask that defines a particular day (by month or week) and time (hour and minute) for the server to check the condition. Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnect

The list of form names the escalation is linked to. The escalation must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

enabled

A flag that specifies whether the escalation is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A query operation performed when the escalation is executed that determines the set of entries to which the escalation actions (defined by the `actionList` parameter) are applied. The system returns `0` (`AR_COND_OP_NONE`) if the escalation has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

ifActionList

The set of actions performed for each entry that matches the criteria defined by the `query` parameter. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseActionList

The set of actions performed if no entries match the criteria defined by the `query` parameter. This list can contain from zero to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The escalation owner.

lastModifiedBy

The user who last modified the escalation.

modifiedDate

The date that the escalation was last modified.

helpText

The help text associated with the escalation. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the escalation.

objPropList

The object properties of the escalation.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetEscalationToXML.

10.4.6 ARGetFieldFromXML

Description

Retrieves information about a field from an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetFieldFromXML(
   ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType fieldName,
   ARNameType appBlockName,
   ARInternalId *fieldID,
   ARFieldMappingStruct *fieldMapping,
   unsigned int *dataType,
   unsigned int *entryMode,
   unsigned int *createMode,
   unsigned int *fieldOption
   ARValueStruct *defaultValue,
   ARPermissionList *permissionList,
   ARFieldLimitStruct *fieldLimit,
   ARDisplayInstanceList *dInstanceList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char **helpText,
   char **changeHistory,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

fieldName

The field name. Each field name must be unique.

appBlockName

For a deployable application, this is the application block name of the field.

Return values

fieldId

The internal ID of the field to retrieve.

fieldMapping

The underlying form that contains the field (only applicable to join forms). Specify `NULL` for this parameter if you do not want to retrieve the field mapping.

dataType

The data type of the field. See the `ARCreateField` information in BMC Remedy AR System C API functions for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve the data type.

entryMode

The entry mode type.

createMode

A flag that indicates the permission status for the field when users submit entries. See `ARCreateField` information in BMC Remedy AR System C API functions for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this flag.

fieldOption

A bitmask that indicates whether the field is to be audited or copied when other fields are audited.

Bit 0:	Value 1: Audit this field. (`AR_FIELD_BITOPTION_AUDIT`)

Bit 1:	Value 1: Copy this field when other fields in the form are audited. (`AR_FIELD_BITOPTION_COPY`)
Bit 2:	Value 1: Indicates this field is for Log Key 1. (`AR_FIELD_BITOPTION_LOG_KEY1`)
Bit 3:	Value 1: Indicates this field is for Log Key 2. (`AR_FIELD_BITOPTION_LOG_KEY2`)
Bits 2 and 3:	Both value 1: Indicates this field is for Log Key 3. (`AR_FIELD_BITOPTION_LOG_KEY3`)

defaultValue

The value to apply if a user submits an entry with no field value (only applicable to data fields). The system returns `0` (`AR_DEFAULT_VALUE_NONE`) if the field has no default. Specify `NULL` for this parameter if you do not want to retrieve the default value.

permissionList

The list of group access permissions.

fieldLimit

The value limits for the field and other properties specific to the field's type. See Defining field limits for a description of the contained structure that applies to the type of field that you are creating. Specify `NULL` (or `AR_FIELD_LIMIT_NONE`) for trim and control fields or if you do not want to retrieve the limits and properties.

dInstanceList

A list of zero or more display properties associated with the field. See ARCreateField for a description of the possible values. The system returns `0` (`AR_DPROP_NONE`) if the field has no display properties. Specify `NULL` for this parameter if you do not want to retrieve this list.

owner

The owner for the field.

lastModifiedBy

The user who last modified the field.

modifiedDate

The date that the field was last modified.

helpText

The help text associated with the field. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the field.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetFieldToXML.

10.4.7 ARGetFilterFromXML

Description

Retrieves information about a filter from an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetFilterFromXML(
   ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType filterName,
   ARNameType appBlockName,
   unsigned int *executionOrder,
   ARWorkflowConnectStruct *workflowConnect,
   unsigned int *executeOn,
   unsigned int *enabled,
   ARQualifierStruct *query,
   ARFilterActionList *ifActionList,
   ARFilterActionList *elseActionList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char **helpText,
   char **changeHistory,
   ARPropList *objPropList,
   unsigned int *errorFilterOptions
   ARNameType errorFilterName
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

filterName

The filter name. Each filter name must be unique.

appBlockName

For a deployable application, this is the application block name of the filter.

Return values

executionOrder

A value between `0` and `1000` (inclusive) that determines the filter execution order. When multiple filters are associated with a form, the value associated with each filter determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnect

The list of form names the filter is linked to. The filter must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

executeOn

A bitmask that indicates the form operations that trigger the filter. Specify `NULL` for this parameter if you do not want to retrieve this value.

enabled

A flag that specifies whether the filter is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A qualification that determines whether the filter is executed. The system returns zero (`AR_COND_OP_NONE`) if the active link has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

ifActionList

The set of actions performed for each entry that matches the criteria defined by the `query` parameter. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseActionList

The set of actions performed if the condition defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owner of the filter.

lastModifiedBy

The user who last modified the filter.

modifiedDate

The date that the filter was last modified.

helpText

The help text associated with the filter. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the filter.

objPropList

The object properties of the filter.

errorFilterOptions

The options for the filter. Returns `AR_FILTER_ERRHANDLER_ENABLE` if an error handler filter is enabled. Set to zero if not.

errorFilterName

The name of error handler filter for this filter; NULL if none. See Error processing for a description of error handlers.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetFilterToXML.

10.4.8 ARGetImageFromXML

Description

Retrieves information about an image from an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetSchemaFromXML(
   ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType imageName,
   ARNameType appBlockName,
   char **imageType,
   unsigned in *contentLength,
   char **checksum,
   ARTimestamp *timestamp,
   char **description,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   char **helpText,
   char **changeHistory,
   ARPropList *objPropList,
   char **imageContent,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The user, sessionId, and server fields are required.

parsedStream

The parsed XML stream.

imageName

The image name. Each schema name must be unique.

appBlockName

For a deployable application, this is the application block name of the image.

Return values

imageType

The type of the image.

contentLength

The image size.

checksum

The checksum of image content.

timestamp

The date that the image was last modified.

description

The image description.

owner

The image owner.

lastModifiedBy

The user who last modified the image.

helpText

The help text associated with the image. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the image.

objPropList

The object properties of the image.

imageContent

The image content.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetImageToXML.

10.4.9 ARGetListXMLObjects

Description

Retrieves object names and types from an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetListXMLObjects(
   ARControlStruct *control,
   ARXMLInputDoc *xmlInputDoc,
   ARUnsignedIntList *objectTypeList,
   ARNameList *objectNameList,
   ARUnsignedIntList *subTypeList,
   ARNameList* *appBlockNameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlDocSource

The XML document output source.

Return values

objectTypeList

A list of the object types.

objectNameList

A list of the object names.

subTypeList

The object subtypes.

appBlockNameList

In a deployable application the list of application block names for the object. The array indexes of the application block name list corresponds to the array index of the **parsedObjects** lists. You can supply NULL for this parameter if you do not want to retrieve it.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

10.4.10 ARGetMenuFromXML

Description

Retrieves information about a menu from an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetMenuFromXML(
ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType menuName,
   ARNameType appBlockName,
   unsigned int *refreshCode,
   ARCharMenuStruct *menuDefn,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char **helpText,
   char **changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

menuName

The menu name. Each menu name must be unique.

appBlockName

For a deployable application, this is the application block name of the menu.

Return values

refreshCode

A value that indicates when the menu is refreshed. See ARCreateCharMenu for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this value.

menuDefn

The definition of the character menu. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The menu owner.

lastModifiedBy

The user who last modified the menu.

modifiedDate

The date that the menu was last modified.

helpText

The help text associated with the menu. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the menu.

objPropList

The object properties of the menu.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

10.4.11 ARGetSchemaFromXML

Description

Retrieves information about a schema (form) from an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetSchemaFromXML(
ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType schemaName,
   ARNameType appBlockName,
   ARCompoundSchema *compoundSchema,
   ARPermissionList *permissionList,
   ARInternalIdList *subAdminGrpList,
   AREntryListFieldList *getListFields,
   ARSortList *sortList,
   ARIndexList *indexList,
   ARArchiveInfoStruct *archiveInfo,
   ARAuditInfoStruct *auditInfo,
   ARNameType defaultVUI,
   ARInternalId *nextFieldID,
   ARULong32 *coreVersion,
   int *upgradeVersion,
   ARFieldInfoList *fieldInfoList,
   ARVuiInfoList *vuiInfoList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char **helpText,
   char **changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

schemaName

The schema name. Each schema name must be unique.

appBlockName

For a deployable application, this is the application block name of the schema (form).

Return values

compoundSchema

The definition of the candidate form or table that contains the candidate field to retrieve.

permissionList

A list of zero or more groups who can access the form. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify `NULL` for this parameter if you do not want to retrieve the group list.

subAdminGrpList

A list of zero or more groups who can administer this form (and the associated filters, escalations, and active links). Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify `NULL` for this parameter if you do not want to retrieve the administrator group list.

getListFields

A list of zero or more fields that identifies the default query list data for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve this value.

sortList

A list of zero or more fields that identifies the default sort order for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve the default sort fields.

indexList

The set of zero or more indexes for the form. Specify `NULL` for this parameter if you do not want to retrieve the index list.

archiveInfo

If the form is to be archived, this is the archive information for the form. Specify `NULL` for this parameter if you do not want to retrieve the archive information. The following are the options for archive type:

`1`	The form is selected to have entries copied to the archive form and deleted from the main form (`AR_ARCHIVE_FORM`).
`2`	The form is selected to have entries deleted from the main form (`AR_ARCHIVE_DELETE`).

16	The form is selected for archive without attachment fields (AR_ARCHIVE_NO_ATTACHMENTS).
32	The form is selected for archive without diary fields (AR_ARCHIVE_NO_DIARY).

In addition to the archive type, **archiveInfo** also stores the form name (if archive type is AR_ARCHIVE_FORM), time to trigger the archive, archive qualification criteria, and name of the main form that is being archived.

auditInfo

If a form is to be audited, this is the audit information for the form. Specify NULL for this parameter if you do not want to retrieve the audit information. These are the values for audit type:

0:	The selected form is not to be audited. (AR_AUDIT_NONE).
1:	Audit fields and copy fields on the selected form are copied to the audit shadow form (AR_AUDIT_COPY).
2:	Audit fields and copy fields on the selected form are copied to the audit log form (AR_AUDIT_LOG).

In addition to the audit type, **auditInfo** also stores the form name (if audit type is AR_AUDIT_COPY or AR_AUDIT_LOG) and the audit qualification criteria.

defaultVui

The label for the default view.

nextFieldID

The next default field identification number, which the server generates if users do not specify it.

coreVersion

The version number of the core fields.

upgradeVersion

The upgrade version for auto generated internal server schemas.

fieldInfoList

The list of schema fields.

vuiInfoList

The list of schema views.

owner

The schema owner.

lastModifiedBy

The user who last modified the schema.

modifiedDate

The date that the schema was last modified.

helpText

The help text associated with the schema. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the schema.

objPropList

The object properties of the schema.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetSchemaToXML.

10.4.12 ARGetVUIFromXML

Description

Retrieves information about a form view from an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetVUIFromXML(
   ARControlStruct *control,
   ARXMLParsedStream *parsedStream,
   ARNameType *schemaName,
   ARNameType appBlockName,
   ARVuiInfoList vuiInfoList,
   ARFieldInfoList fieldInfoList,
   ARTimestamp *modifiedDate,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

schemaName

The name of the view. Each view name must be unique.

appBlockName

For a deployable application, this is the application block name of the view.

Return values

vuiInfoList

The list of schema views.

fieldInfoList

The list of view fields.

modifiedDate

The date that the view was last modified.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetVUIToXML.

10.4.13 ARParseXMLDocument

Description

Parses the contents of an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARParseXMLDocument(
   ARControlStruct *control,
   ARXMLInputDoc *xmlnputDoc,
   ARStructItemList *objectsToParse,
   ARXMLParsedStream *parsedStream,
   ARStructItemList *parsedObjects,
   ARNameList *appBlockNameList,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlInputDoc

The XML document to be parsed.

objectsToParse

The specific list of objects to parse. Specify `NULL` if you do not want to parse any objects.

Return values

parsedStream

The parsed XML stream.

parsedObjects

The list of parsed objects.

appBlockNameList

In a deployable application the list of application block names for the object. The array indexes of the application block name list corresponds to the array index of the **parsedObjects** lists. You can supply `NULL` for this parameter if you do not want to retrieve it.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

10.4.14 ARSetActiveLinkToXML

Description

Saves information about an active link in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetActiveLinkToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType activeLinkName,
   unsigned int *executionOrder,
   ARWorkflowConnectStruct *workflowConnect,
   ARInternalIdList *accessList,
   unsigned int *executeOn,
   ARInternalId *controlFieldID,
   ARInternalId *focusFieldID,
   unsigned int *enabled,
   ARQualifierStruct *query,
   ARActiveLinkActionList *ifActionList,
   ARActiveLinkActionList *elseActionList,
   ARSupportFileInfoList *supportFileList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char *helpText,
   char *changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   unsigned int *errorActlinkOptions,
   ARNameType errorActlinkName,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

activeLinkName

The name of the active link to be set.

executionOrder

A value between `0` and `1000` (inclusive) that determines the active link execution order. When multiple active links are associated with a form, the value associated with each active link determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnect

The list of form names the active link is linked to. The active link must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

accessList

A list of lists containing zero or more groups who can access the fields retrieved. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify `NULL` for this parameter if you do not want to retrieve the permissions.

executeOn

A bitmask that indicates the form operations that trigger the active link. Specify `NULL` for this parameter if you do not want to retrieve this value.

controlFieldID

The ID of the field that represents the button, toolbar button, or menu item associated with executing the active link. The system returns zero if the **executeMask** does not include the `AR_EXECUTE_ON_BUTTON` condition. Specify `NULL` for this parameter if you do not want to retrieve this value.

focusFieldID

The ID of the field associated with executing the active link by pressing Return or selecting a character menu item. The system returns zero if the **executeMask** does not include the `AR_EXECUTE_ON_RETURN` or `AR_EXECUTE_ON_MENU_CHOICE` conditions. Specify `NULL` for this parameter if you do not want to retrieve this value.

enabled

A flag that specifies whether the active link is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A query operation performed when the escalation is executed that determines the set of entries to which the escalation actions (defined by the **actionList** parameter) are applied. The system returns `0` (`AR_COND_OP_NONE`) if the escalation has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

ifActionList

The set of actions performed for each entry that matches the criteria defined by the `query` parameter. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseActionList

The set of actions performed if the condition defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

supportFileList

The support file that the active link uses.

owner

The active link owner.

lastModifiedBy

The user who last modified the active link.

modifiedDate

The date that the active link was last modified.

helpText

The help text associated with the active link. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the active link.

objPropList

The object properties of the active link.

arDocVersion

The XML document version.

errorActlinkOptions

Reserved for future use. Set to `NULL`.

errorActlinkName

Reserved for future use. Set to `NULL`.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetActiveLinkFromXML.

10.4.15 ARSetContainerToXML

Description

Saves information about a container in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetContainerToXML(
ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType containerName,
   ARPermissionList *permissionList,
   ARInternalIdList *subAdminGrpList,
   ARContainerOwnerObjList *ownerObjectList,
   char *label,
   char *description,
   unsigned int *containerType,
   ARReferenceList *referenceList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char *helpText,
   char *changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

containerName

The container name. Each container name must be unique.

permissionList

A list of zero or more groups who can access this container. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify `NULL` for this parameter if you do not want to retrieve this value.

subAdminGrpList

The list of groups that have access to the container.

ownerObjectList

A list of schemas that own this container. This parameter can be `NULL` if the container exists globally.

label

The label for this container. It can be as many as 255 characters long or `NULL`.

description

The description for this container. It can be as many as 2000 characters long or `NULL`.

containerType

The type for this container — either guide (`ARCON_GUIDE`), application (`ARCON_APP`), or a custom type that you have defined.

referenceList

A list of pointers to the objects referenced by this container. References can be to internal BMC Remedy AR System objects (for example, guides reference active links and applications reference forms) or to external objects such as URLs or file names. Specify `NULL` for this parameter if you do not want to associate any objects with this container.

owner

The container owner.

lastModifiedBy

The user who last modified the container.

modifiedDate

The date that the container was last modified.

helpText

The help text associated with the container. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the container.

objPropList

The object properties of the container.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetContainerFromXML.

10.4.16 ARSetDSOMappingToXML

Description

Saves information about a DSO mapping in an XML document. For more information about DSO mapping, see Enabling distributed mappings.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetDSOMappingToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType mappingName,
   ARNameType fromSchema,
   ARServerNameType fromServer,
   ARNameType toSchema,
   ARServerNameType toServer,
   unsigned int *enabled,
   unsigned int updateCode,
   unsigned int *transferMode,
   unsigned int *mapType,
   unsigned int *rtnMapType,
   unsigned int *defaultMap,
   unsigned int *duplicateAction,
   unsigned int *patternMatch,
   unsigned int *requiredFields,
   long *retryTime,
   char **mapping,
   char **rtnMapping,
   char **matchQual,
   char **excludeFlds
   char **rtnExcludeFlds
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char *helpText,
   char *changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

mappingName

The name of the schema to be exported. Each schema name must be unique.

fromSchema

The name of the source schema. Each schema name must be unique.

fromServer

The name of the source server.

toSchema

The name of the target schema.

toServer

The name of the target server.

enabled

A flag that specifies whether the DSO pool is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

updateCode

A value that indicates when the server updates the entry after the entry is made.

transferMode

The transfer mode.

mapType

The type of mapping to use on transfer.

rtnMapType

The type of mapping to use on update or return.

defaultMap

A value that indicates the default mapping.

duplicateAction

The action to perform on transfer to the existing entry.

patternMatch

A value that indicates that pattern matching is enabled.

requiredField

A value that indicates if the required fields can be ignored.

retryTime

The maximum number of seconds that the server waits to remap the schema, before it cancels the operation.

mapping

A value that indicates custom mapping is turned on, if the `mapType` parameter is set to `custom`.

rtnMapping

A value that indicates custom mapping is turned on, if the `rtnMapType` parameter is set to `custom`.

matchQual

The matching qualification.

excludeFlds

A value that indicates the fields to be excluded from transfer mapping when the `mapType` value is Like IDs or Like Names. The list of field ids must be separated by a semi-colon (;).

rtnExcludeFlds

A value that indicates the fields to be excluded from update or return mapping when the rtnMapType value is Like IDs or Like Names. The list of field ids must be separated by a semi-colon (;).

owner

The owner for the DSO mapping.

lastModifiedBy

The user who last modified the DSO mapping.

modifiedDate

The date that the DSO mapping was last modified.

helpText

The help text associated with the DSO mapping. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the DSO mapping.

objPropList

The properties of the DSO mapping.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

10.4.17 ARSetDSOPoolToXML

Description

Saves information about a DSO pool in an XML document. For more information about DSO mapping, see Enabling distributed mappings.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetDSOPoolToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType poolName,
   unsigned int *enabled,
   unsigned int *defaultPool,
   long *threadCount,
   char **connection,
   unsigned int *polling,
   unsigned int *pollingInterval,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char *helpText,
   char *changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

poolName

The name of the DSO pool. Each pool name must be unique.

enabled

A flag that specifies whether the DSO pool is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

defaultPool

The code that indicates whether there is a default pool.

threadcount

The thread count of the DSO pool.

connection

This parameter is reserved for future use.

polling

Indicates whether this pool polls the pending queue for pending items assigned to it. Pools are signaled, by default.

pollingInterval

The polling interval (if the pool is a polling pool).

owner

The owner for the DSO pool.

lastModifiedBy

The user who last modified the DSO pool.

modifiedDate

The date that the DSO pool was last modified.

helpText

The help text associated with the DSO pool. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the DSO pool.

objPropList

The properties of the DSO pool.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

10.4.18 ARSetEscalationToXML

Description

Saves information about an escalation in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARsetEscalationToXML(
ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType escalationName,
   AREscalationTmStruct *escalationTime,
   ARWorkflowConnectStruct *workflowConnect,
   unsigned int *enabled,
   ARQualifierStruct *query,
   ARFilterActionList *ifActionList,
   ARFilterActionList *elseActionList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char *helpText,
   char *changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

escalationName

The name of the escalation. Each escalation name must be unique.

escalationTime

The time specification for evaluating the escalation condition. This parameter can take one of two forms: a time interval that defines how frequently the server checks the escalation condition (in seconds) or a bitmask that defines a particular day (by month or week) and time (hour and minute) for the server to check the condition. Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnect

The list of form names the escalation is linked to. The escalation must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

enabled

A flag that specifies whether the escalation is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A query operation performed when the escalation is executed that determines the set of entries to which the escalation actions (defined by the `actionList` parameter) are applied. The system returns `0` (

AR_COND_OP_NONE) if the escalation has no qualification. Specify NULL for this parameter if you do not want to retrieve this value.

ifActionList

The set of actions performed for each entry that matches the criteria defined by the query parameter. This list can contain from 1 to 25 actions (limited by AR_MAX_ACTIONS). Specify NULL for this parameter if you do not want to retrieve this value.

elseActionList

The set of actions performed if no entries match the criteria defined by the query parameter. This list can contain from zero to 25 actions (limited by AR_MAX_ACTIONS). Specify NULL for this parameter if you do not want to retrieve this value.

owner

The escalation owner.

lastModifiedBy

The user who last modified the escalation.

modifiedDate

The date that the escalation was last modified.

helpText

The help text associated with the escalation. Specify NULL for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the escalation.

objPropList

The object properties of the escalation.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetEscalationFromXML.

10.4.19 ARSetFieldToXML

Description

Saves information about a field in an XML file.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetFieldToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType fieldName,
   ARInternalId *fieldID,
   ARFieldMappingStruct *fieldMapping,
   unsigned int *dataType,
   unsigned int *entryMode,
   unsigned int *createMode,
   unsigned int *fieldOption
   ARValueStruct *defaultValue,
   ARPermissionList *permissionList,
   ARFieldLimitStruct *fieldLimit,
   ARDisplayInstanceList *dInstanceList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char *helpText,
   char *changeHistory,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

fieldName

The field name. Each field name must be unique.

fieldId

The internal ID of the field to retrieve.

fieldMapping

The underlying form that contains the field (only applicable to join forms). Specify `NULL` for this parameter if you do not want to retrieve the field mapping.

dataType

The data type of the field. See ARCreateField for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve the data type.

entryMode

The entry mode type.

createMode

A flag that indicates the permission status for the field when users submit entries. See ARCreateField for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this flag.

fieldOption

A bitmask that indicates whether the field is to be audited or copied when other fields are audited.

Bit 0:	Value 1: Audit this field. (`AR_FIELD_BITOPTION_AUDIT`)
Bit 1:	Value 1: Copy this field when other fields in the form are audited. (`AR_FIELD_BITOPTION_COPY`)
Bit 2:	Value 1: Indicates this field is for Log Key 1. (`AR_FIELD_BITOPTION_LOG_KEY1`)
Bit 3:	Value 1: Indicates this field is for Log Key 2. (`AR_FIELD_BITOPTION_LOG_KEY2`)
Bits 2 and 3:	Both value 1: Indicates this field is for Log Key 3. (`AR_FIELD_BITOPTION_LOG_KEY3`)

defaultValue

The value to apply if a user submits an entry with no field value (only applicable to data fields). The system returns `0` (`AR_DEFAULT_VALUE_NONE`) if the field has no default. Specify `NULL` for this parameter if you do not want to retrieve the default value.

permissionList

The list of group access permissions.

fieldLimit

The value limits for the field and other properties specific to the field's type. See Defining field limits for a description of the contained structure that applies to the type of field that you are creating. Specify `NULL` (or `AR_FIELD_LIMIT_NONE`) for trim and control fields or if you do not want to retrieve the limits and properties.

dInstanceList

A list of zero or more display properties associated with the field. See ARCreateField for a description of the possible values. The system returns `0` (`AR_DPROP_NONE`) if the field has no display properties. Specify `NULL` for this parameter if you do not want to retrieve this list.

owner

The owner for the field.

lastModifiedBy

The user who last modified the field.

modifiedDate

The date that the field was last modified.

helpText

The help text associated with the field. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the field.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetFieldFromXML.

10.4.20 ARSetFilterToXML

Description

Saves information about a filter in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetFilterToXML(
ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType filterName,
   unsigned int *executionOrder,
   ARWorkflowConnectStruct *workflowConnect,
   unsigned int *executeOn,
   unsigned int *enabled,
   ARQualifierStruct *query,
   ARFilterActionList *ifActionList,
   ARFilterActionList *elseActionList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char *helpText,
   char *changeHistory,
   ARPropList *objPropList,
   unsigned int *errorFilterOptions
   ARNameType errorFilterName
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

filterName

The filter name. Each filter name must be unique.

executionOrder

A value between `0` and `1000` (inclusive) that determines the filter execution order. When multiple filters are associated with a form, the value associated with each filter determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnect

The list of form names the filter is linked to. The filter must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

executeOn

A bitmask that indicates the form operations that trigger the filter. Specify `NULL` for this parameter if you do not want to retrieve this value.

enabled

A flag that specifies whether the filter is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A qualification that determines whether the filter is executed. The system returns zero (`AR_COND_OP_NONE`) if the filter has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

ifActionList

The set of actions performed for each entry that matches the criteria defined by the `query` parameter. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseActionList

The set of actions performed if the condition defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owner of the filter.

lastModifiedBy

The user who last modified the filter.

modifiedDate

The date that the filter was last modified.

helpText

The help text associated with the filter. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the filter.

objPropList

The object properties of the filter.

errorFilterOptions

The error handler options for the filter. Set to `AR_FILTER_ERRHANDLER_ENABLE` to enable an error handler filter. Set to zero to disable.

errorFilterName

The name of error handler filter for this filter; NULL if none.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetFilterFromXML.

10.4.21 ARSetImageToXML

Description

Saves information about an image to an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetSchemaToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType imageName,
   char *imageType,
   char *description
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   char *helpText,
   char *changeHistory,
   ARPropList *objPropList,
   char checksum,
   ARTimestamp *modifiedDate,
   ARImageDataStruct imageContent,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

imageName

The image name. Each image name must be unique.

imageType

The image type.

description

The image description.

owner

The image owner.

lastModifiedBy

The user who last modified the image.

helpText

The help text associated with the image. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the image.

objPropList

The object properties of the image.

checksum

The checksum of the image content.

modifiedDate

The date that the image was last modified.

imageContent

The image content.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetImageFromXML.

10.4.22 ARSetMenuToXML

Description

Save information about a menu in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetMenuToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType menuName,
   unsigned int *refreshCode,
   ARCharMenuStruct *menuDefn,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char *helpText,
   char *changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

menuName

The menu name. Each menu name must be unique.

refreshCode

A value that indicates when the menu is refreshed. See ARCreateCharMenu for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this value.

menuDefn

The definition of the character menu. Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The menu owner.

lastModifiedBy

The user who last modified the menu.

modifiedDate

The date that the menu was last modified.

helpText

The help text associated with the menu. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the menu.

objPropList

The object properties of the menu.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetMenuFromXML.

10.4.23 ARSetSchemaToXML

Description

Saves information about a schema (form) in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetSchemaToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrFlag,
   ARNameType schemaName,
   ARCompoundSchema *compoundSchema,
   ARPermissionList *permissionList,
   ARInternalIdList *subAdminGrpList,
   AREntryListFieldList *getListFields,
   ARSortList *sortList,
   ARIndexList *indexList,
   ARArchiveInfoStruct *archiveInfo,
   ARAuditInfoStruct *auditInfo,
   ARNameType *defaultVUI,
   ARInternalId *nextFieldID,
   ARULong32 *coreVersion,
   int *upgradeVersion,
   ARFieldInfoList *fieldInfoList,
   ARVuiInfoList *vuiInfoList,
   ARAccessNameType owner,
   ARAccessNameType lastModifiedBy,
   ARTimestamp *modifiedDate,
   char *helpText,
   char *changeHistory,
   ARPropList *objPropList,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

schemaName

The schema name. Each schema name must be unique.

compoundSchema

The definition of the candidate form or table that contains the candidate field to retrieve.

permissionList

A list of zero or more groups who can access the form. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify `NULL` for this parameter if you do not want to retrieve the group list.

subAdminGrpList

A list of zero or more groups who can administer this form (and the associated filters, escalations, and active links). Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify `NULL` for this parameter if you do not want to retrieve the administrator group list.

getListField

A list of zero or more fields that identifies the default query list data for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve this value.

sortList

A list of zero or more fields that identifies the default sort order for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve the default sort fields.

indexList

The set of zero or more indexes for the form. Specify `NULL` for this parameter if you do not want to retrieve the index list.

archiveInfo

If the form is to be archived, this is the archive information for the form. Specify `NULL` for this parameter if you do not want to create or set the archive information. These are the options for archive type:

`1`:	The form is selected to have entries copied and deleted to the archive form (`AR_ARCHIVE_FORM`).
`2`:	The form is selected to have entries deleted from the main form (`AR_ARCHIVE_DELETE`).
`32`:	The form is selected for archive but without attachment field (`AR_ARCHIVE_NO_ATTACHMENTS`).
`64`:	The form is selected for archive without diary fields (`AR_ARCHIVE_NO_DIARY`).

In addition to the archive type, `archiveInfo` also stores the form name (if archive type is `AR_ARCHIVE_FORM`), time to trigger the archive, archive qualification criteria, and name of the main form that is being archived.

auditInfo

If a form is to be audited, this is the audit information for the form. Specify `NULL` for this parameter if you do not want to create or set the audit information. These are the values for audit type:

`0`:	The selected form is not to be audited. (`AR_AUDIT_NONE`).
`1`:	Audit fields and copy fields on the selected form are copied to the audit shadow form (`AR_AUDIT_COPY`).
`2`:	Audit fields and copy fields on the selected form are copied to the audit log form (`AR_AUDIT_LOG`).

In addition to the audit type, `auditInfo` also stores the form name (if audit type is `AR_AUDIT_COPY` or `AR_AUDIT_LOG`) and the audit qualification criteria.

defaultVui

The label for the default view.

nextFieldID

The next default field identification number, which the server generates if users do not specify it.

coreVersion

The version number of the core fields.

upgradeVersion

The upgrade version for auto generated internal server schemas. To disable this option, set it to zero (`0`).

fieldInfoList

The list of schema fields.

vuiInfoList

The list of schema views.

owner

The schema owner.

lastModifiedBy

The user who last modified the schema.

modifiedDate

The date that the schema was last modified.

helpText

The help text associated with the schema. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the schema.

objPropList

The object properties of the schema.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetSchemaFromXML.

10.4.24 ARSetVUIToXML

Description

Saves information about a form view in an XML document.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetVUIToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARBoolean xmlDocHdrFtrMask,
   ARNameType schemName,
   ARVuiInfoList *vuiInfoList,
   ARFieldInfoList *fieldInfoList,
   ARTimestamp *modifiedDate,
   unsigned int *arDocVersion,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrMask

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

schemaName

The name of the view. Each view name must be unique.

vuiInfoList

The list of schema views.

fieldInfoList

The list of view fields.

modifiedDate

The date that the view was last modified.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetVUIFromXML.

10.4.25 ARSetXMLDocFooterToXML

Description

Converts document footers to XML.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetXMLDocFooterToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetXMLDocHeaderToXML.

10.4.26 ARSetXMLDocHeaderToXML

Description

Converts document headers to XML.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetXMLDocHeaderToXML(
   ARControlStruct *control,
   ARXMLOutputDoc *xmlOutputDoc,
   ARStatusList *status)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetXMLDocFooterToXML.

10.4.27 ARGetSchemaWithExtensionsFromXML

Description

Retrieves information about the form with the indicated name from the specified server. This information does not include the form's field definitions.

Privileges

All users.

Synopsis

```
int ARGetSchemaWithExtensionsFromXML(
ARControlStruct         *control,
ARNameType               schemaName,
ARNameType               appBlockName,
ARCompoundSchema        *compoundSchema,
ARPermissionList        *permissionList,
ARPermissionList        *addedPermissionList,
ARInternalIdList        *subAdminGrpList,
ARInternalIdList        *addedSubAdminGrpList,
AREntryListFieldList    *getListFields,
ARSortList              *sortList,
ARIndexList             *indexList,
ARIndexList             *addedIndexList,
ARArchiveInfoStruct     *archiveInfo,
ARAuditInfoStruct       *auditInfo,
ARNameType               defaultVUI,
ARInternalId            *nextFieldID,
ARULong32               *coreVersion,
int                     *upgradeVersion,
ARExportFieldInfoList   *fieldInfoList,
ARVuiInfoList           *vuiInfoList,
ARAccessNameType         owner,
ARAccessNameType         lastModifiedBy,
ARTimestamp             *modifiedDate,
char                   **helpText,
char                   **changeHistory,
ARPropList              *objPropList,
unsigned int            *arDocVersion,
ARStatusList            *status
)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The user, sessionId, and server fields are required.

schemaName

Name of the schema. Each schema name must be unique.

appBlockName

For a deployable application, this is the application block name of the schema (form).

Return values

compoundSchema

The definition of the candidate form or table that contains the candidate field to retrieve.

permissionList

A list of zero or more groups who can access the form. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify NULL for this parameter if you do not want to retrieve the group list.

addedPermissionList

The addedPermissionList is a set of groups that is added to the permission list when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

subAdminGrpList

A list of zero or more groups who can administer this form (and the associated filters, escalations, and active links). Users must belong to both a specified group and the Subadministrator group to obtain these privileges. Specify NULL for this parameter if you do not want to retrieve the administrator group list.

addedSubAdminGrpList

The addedSubAdminGrpList is a set of groups that is added to the subadministrator group list when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

getListFields

A list of zero or more fields that identifies the default query list data for retrieving form entries. Specify NULL for this parameter if you do not want to retrieve this value.

sortList

A list of zero or more fields that identifies the default sort order for retrieving form entries. Specify NULL for this parameter if you do not want to retrieve the default sort fields.

indexList

The set of zero or more indexes for the form. Specify NULL for this parameter if you do not want to retrieve the index list.

addedIndexList

A set of groups that is added to the index list when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

archiveInfo

If the form is to be archived, this is the archive information for the form. Specify `NULL` for this parameter if you do not want to retrieve the archive information. The following are the options for archive type:

`1`	The form is selected to have entries copied to the archive form and deleted from the main form (`AR_ARCHIVE_FORM`).
`2`	The form is selected to have entries deleted from the main form (`AR_ARCHIVE_DELETE`).
`16`	The form is selected for archive without attachment fields (`AR_ARCHIVE_NO_ATTACHMENTS`).
`32`	The form is selected for archive without diary fields (`AR_ARCHIVE_NO_DIARY`).

In addition to the archive type, **archiveInfo** also stores the form name (if archive type is `AR_ARCHIVE_FORM`), time to trigger the archive, archive qualification criteria, and name of the main form that is being archived.

auditInfo

If a form is to be audited, this is the audit information for the form. Specify `NULL` for this parameter if you do not want to retrieve the audit information. These are the values for audit type:

`0` :	The selected form is not to be audited. (`AR_AUDIT_NONE`).
`1` :	Audit fields and copy fields on the selected form are copied to the audit shadow form (`AR_AUDIT_COPY`).
`2` :	Audit fields and copy fields on the selected form are copied to the audit log form (`AR_AUDIT_LOG`).

In addition to the audit type, **auditInfo** also stores the form name (if audit type is `AR_AUDIT_COPY` or `AR_AUDIT_LOG`) and the audit qualification criteria.

defaultVui

The label for the default view.

nextFieldID

The next default field identification number, which the server generates if users do not specify it.

coreVersion

The version number of the core fields.

upgradeVersion

The upgrade version for auto generated internal server schemas.

fieldInfoList

The list of schema fields.

vuiInfoList

The list of schema views.

owner

The schema owner.

lastModifiedBy

The user who last modified the schema.

modifiedDate

The date that the schema was last modified.

helpText

The help text associated with the schema. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the schema.

objPropList

The object properties of the schema.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetSchemaFromXML

10.4.28 ARGetFilterWithExtensionsFromXML

Description

Retrieves information about the specified filter from the specified server.

Privileges

All users.

Synopsis

```
int ARGetFilterWithExtensionsFromXML(
ARControlStruct         *control,
ARXMLParsedStream       *parsedStream,
ARNameType               filterName,
ARNameType               appBlockName,
unsigned int            *executionOrder,
ARWorkflowConnectStruct *workflowConnect,
ARWorkflowConnectStruct *addedWorkflowConnect,
unsigned int            *executeOn,
unsigned int            *enabled,
ARQualifierStruct       *query,
ARFilterActionList      *ifActionList,
ARFilterActionList      *elseActionList,
ARAccessNameType         owner,
ARAccessNameType         lastModifiedBy,
ARTimestamp             *modifiedDate,
char                   **helpText,
char                   **changeHistory,
ARPropList              *objPropList,
unsigned int            *errorHandlerOptions,
ARNameType               errorHandlerName,
unsigned int            *arDocVersion,
ARStatusList            *status
)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

filterName

The filter name. Each filter name must be unique.

appBlockName

For a deployable application, this is the application block name of the filter.

Return values

executionOrder

A value between `0` and `1000` (inclusive) that determines the filter execution order. When multiple filters are associated with a form, the value associated with each filter determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnect

The list of form names the filter is linked to. The filter must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

addedWorkflowConnect

A set of groups that is added to the form names when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

executeOn

A bitmask that indicates the form operations that trigger the filter. Specify `NULL` for this parameter if you do not want to retrieve this value.

enabled

A flag that specifies whether the filter is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A qualification that determines whether the filter is executed. The system returns zero (`AR_COND_OP_NONE`) if the active link has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

ifActionList

The set of actions performed for each entry that matches the criteria defined by the `query` parameter. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseActionList

The set of actions performed if the condition defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The owner of the filter.

lastModifiedBy

The user who last modified the filter.

modifiedDate

The date that the filter was last modified.

helpText

The help text associated with the filter. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the filter.

objPropList

The object properties of the filter.

errorFilterOptions

The options for the filter. Returns `AR_FILTER_ERRHANDLER_ENABLE` if an error handler filter is enabled. Set to zero if not.

errorFilterName

The name of error handler filter for this filter; NULL if none. See Error processing for a description of error handlers.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetFilterFromXML

10.4.29 ARGetActiveLinkWithExtensionsFromXML

Description

Retrieves information about the active link with the indicated name from the specified server.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetActiveLinkWithExtensionsFromXML(
ARControlStruct         *control,
ARXMLParsedStream       *parsedStream,
ARNameType               activeLinkName,
ARNameType               appBlockName,
unsigned int            *executionOrder,
ARWorkflowConnectStruct *workflowConnect,
ARWorkflowConnectStruct *addedWorkflowConnect,
ARInternalIdList        *accessList,
ARInternalIdList        *addedAccessList,
unsigned int            *executeOn,
ARInternalId            *controlFieldID,
ARInternalId            *focusFieldID,
unsigned int            *enabled,
ARQualifierStruct       *query,
ARActiveLinkActionList  *ifActionList,
ARActiveLinkActionList  *elseActionList,
ARSupportFileInfoList   *supportFileList,
ARAccessNameType         owner,
ARAccessNameType         lastModifiedBy,
ARTimestamp             *modifiedDate,
char                   **helpText,
char                   **changeHistory,
ARPropList              *objPropList,
unsigned int            *errorHandlerOptions,
ARNameType               errorHandlerName,
unsigned int            *arDocVersion,
ARStatusList            *status
)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

activeLinkName

The active link name. Each active link name must be unique.

appBlockName

For deployable applications, the application block name of the active link.

Return values

executionOrder

A value between `0` and `1000` (inclusive) that determines the active link execution order. When multiple active links are associated with a form, the value associated with each active link determines the order in which they are processed (from lowest to highest). Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnect

The list of form names the active link is linked to. The active link must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

addedWorkflowConnect

A set of groups that is added to the form names when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

accessList

A list of lists containing zero or more groups who can access the fields retrieved. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify `NULL` for this parameter if you do not want to retrieve the permissions.

addedAccessList

A set of groups that is added to the field list when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

executeOn

A bitmask that indicates the form operations that trigger the active link. Specify `NULL` for this parameter if you do not want to retrieve this value.

controlFieldID

The ID of the field that represents the button, toolbar button, or menu item associated with executing the active link. The system returns zero if the **executeMask** does not include the `AR_EXECUTE_ON_BUTTON` condition. Specify `NULL` for this parameter if you do not want to retrieve this value.

focusFieldID

The ID of the field associated with executing the active link by pressing Return or selecting a character menu item. The system returns zero if the **executeMask** does not include the `AR_EXECUTE_ON_RETURN` or `AR_EXECUTE_ON_MENU_CHOICE` conditions. Specify `NULL` for this parameter if you do not want to retrieve this value.

enabled

A flag that specifies whether the active link is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A qualification that determines whether the active link is executed. The system returns zero (`AR_COND_OP_NONE`) if the active link has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

ifActionList

The set of actions performed if the condition defined by the `query` parameter is satisfied. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseActionList

The set of actions performed if the condition defined by the `query` parameter is not satisfied. This list can contain from 0 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

supportFileList

The support file that the active link uses.

owner

The active link owner. Specify `NULL` for this parameter if you do not want to retrieve this value.

lastModifiedBy

The user who last modified the active link. Specify `NULL` for this parameter if you do not want to retrieve this value.

modifiedDate

The date that the active link was last modified.

helpText

The help text associated with the active link. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the active link.

objPropList

The object properties of the active link.

arDocVersion

The XML document version.

errorActlinkOptions

Reserved for future use. Set to `NULL`.

errorActlinkName

Reserved for future use. Set to `NULL`.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetActiveLinkFromXML

10.4.30 ARGetEscalationWithExtensionsFromXML

Description

Retrieves information about the escalation with the indicated name from the specified server.

Privileges

All users.

Synopsis

```
#include "arextern.h"
```

```
int ARGetEscalationWithExtensionsFromXML(
ARControlStruct         *control,
ARXMLParsedStream       *parsedStream,
ARNameType               escalationName,
ARNameType               appBlockName,
AREscalationTmStruct    *escalationTime,
ARWorkflowConnectStruct *workflowConnect,
ARWorkflowConnectStruct *addedWorkflowConnect,
unsigned int            *enabled,
ARQualifierStruct       *query,
ARFilterActionList      *ifActionList,
ARFilterActionList      *elseActionList,
ARAccessNameType         owner,
ARAccessNameType         lastModifiedBy,
ARTimestamp             *modifiedDate,
char                   **helpText,
char                   **changeHistory,
ARPropList              *objPropList,
unsigned int            *arDocVersion,
ARStatusList            *status
)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

escalationName

The name of the escalation. Each escalation name must be unique.

appBlockName

For a deployable application, this is the application block name of the escalation.

Return values

escalationTime

The time specification for evaluating the escalation condition. This parameter can take one of two forms: a time interval that defines how frequently the server checks the escalation condition (in seconds) or a bitmask that defines a particular day (by month or week) and time (hour and minute) for the server to check the condition. Specify `NULL` for this parameter if you do not want to retrieve this value.

workflowConnect

The list of form names the escalation is linked to. The escalation must be associated with a single form or a list of forms that currently exists on the server. Specify `NULL` for this parameter if you do not want to retrieve this value.

addedWorkflowConnect

A set of groups that is added to the form names when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

enabled

A flag that specifies whether the escalation is disabled (`0`) or enabled (`1`). Specify `NULL` for this parameter if you do not want to retrieve this value.

query

A query operation performed when the escalation is executed that determines the set of entries to which the escalation actions (defined by the `actionList` parameter) are applied. The system returns `0` (`AR_COND_OP_NONE`) if the escalation has no qualification. Specify `NULL` for this parameter if you do not want to retrieve this value.

ifActionList

The set of actions performed for each entry that matches the criteria defined by the `query` parameter. This list can contain from 1 to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

elseActionList

The set of actions performed if no entries match the criteria defined by the `query` parameter. This list can contain from zero to 25 actions (limited by `AR_MAX_ACTIONS`). Specify `NULL` for this parameter if you do not want to retrieve this value.

owner

The escalation owner.

lastModifiedBy

The user who last modified the escalation.

modifiedDate

The date that the escalation was last modified.

helpText

The help text associated with the escalation. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the escalation.

objPropList

The object properties of the escalation.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetEscalationFromXML

10.4.31 ARGetContainerWithExtensionsFromXML

Description

Retrieves the contents of the container with the indicated name from the specified server. It can return references of a single, specified type, of all types, or of an exclude reference type. The system returns information for accessible references and does nothing for references for which the user does not have access.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetContainerWithExtensionsFromXML(
ARControlStruct         *control,
ARXMLParsedStream       *parsedStream,
ARNameType               containerName,
ARNameType               appBlockName,
ARPermissionList        *permissionList,
ARPermissionList        *addedPermissionList,
ARInternalIdList        *subAdminGrpList,
ARInternalIdList        *addedSubAdminGrpList,
ARContainerOwnerObjList *ownerObjectList,
ARContainerOwnerObjList *addedOwnerObjectList,
char                   **label,
char                   **description,
unsigned int            *containerType,
ARReferenceList         *referenceList,
ARAccessNameType         owner,
ARAccessNameType         lastModifiedBy,
ARTimestamp             *modifiedDate,
char                   **helpText,
char                   **changeHistory,
ARPropList              *objPropList,
unsigned int            *arDocVersion,
ARStatusList            *status
)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

containerName

The container name. Each container name must be unique.

appBlockName

For a deployable application, this is the application block name of the container.

Return values

permissionList

A list of zero or more groups who can access this container. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify `NULL` for this parameter if you do not want to retrieve this value.

addedPermissionList

A set of groups that is added to the permission list when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

subAdminGrpList

The list of groups that have access to the container.

addedSubAdminGrpList

A set of groups that is added to the subadministrator group list when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

ownerObjectList

A list of schemas that own this container. This parameter can be `NULL` if the container exists globally.

addedOwnerObjectList

A set of groups that is added to the owner object group list when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

label

The label for this container. It can be as many as 255 characters long or `NULL`.

description

The description for this container. It can be as many as 2000 characters long or `NULL`.

containerType

The type for this container — either guide (`ARCON_GUIDE`), application (`ARCON_APP`), or a custom type you have defined.

referenceList

A list of pointers to the objects referenced by this container. References can be to internal BMC Remedy AR System objects (for example, guides reference active links and applications reference forms) or to external objects such as URLs or file names. Specify **NULL** for this parameter if you do not want to associate any objects with this container.

owner

The container owner.

lastModifiedBy

The user who last modified the container.

modifiedDate

The date that the container was last modified.

helpText

The help text associated with the container. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the container.

objPropList

The object properties of the container.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetContainerFromXML

10.4.32 ARGetVUIWithExtensionsFromXML

Description

Retrieves information about the form view (VUI) with the indicated ID from the specified server.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetVUIWithExtensionsFromXML(
ARControlStruct         *control,
ARXMLParsedStream       *parsedStream,
ARNameType               schemaName,
ARNameType               appBlockName,
ARVuiInfoList           *vuiInfoList,
ARExportFieldInfoList   *fieldInfoList,
ARTimestamp             *modifiedDate,
unsigned int            *arDocVersion,
ARStatusList            *status
)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

schemaName

The name of the view. Each view name must be unique.

appBlockName

For a deployable application, this is the application block name of the view.

Return values

vuiInfoList

The list of schema views.

fieldInfoList

The list of view fields.

modifiedDate

The date that the view was last modified.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetVUIFromXML

10.4.33 ARGetFieldWithExtensionsFromXML

Description

Retrieves information about the form field with the indicated ID from the specified server.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARGetFieldWithExtensionsFromXML(
ARControlStruct         *control,
ARXMLParsedStream       *parsedStream,
ARNameType               fieldName,
ARNameType               appBlockName,
ARInternalId            *fieldID,
ARFieldMappingStruct    *fieldMapping,
unsigned int            *dataType,
unsigned int            *entryMode,
unsigned int            *createMode,
unsigned int            *fieldOption,
ARValueStruct           *defaultValue,
ARPermissionList        *permissionList,
ARPermissionList        *addedPermissionList,
ARFieldLimitStruct      *fieldLimit,
ARDisplayInstanceList   *dInstanceList,
ARAccessNameType         owner,
ARAccessNameType         lastModifiedBy,
ARTimestamp             *modifiedDate,
char                   **helpText,
char                   **changeHistory,
unsigned int            *arDocVersion,
ARStatusList            *status
)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

parsedStream

The parsed XML stream.

fieldName

The field name. Each field name must be unique.

appBlockName

For a deployable application, this is the application block name of the field.

Return values

fieldId

The internal ID of the field to retrieve.

fieldMapping

The underlying form that contains the field (only applicable to join forms). Specify `NULL` for this parameter if you do not want to retrieve the field mapping.

dataType

The data type of the field. See the `ARCreateField` information in BMC Remedy AR System C API functions for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve the data type.

entryMode

The entry mode type.

createMode

A flag that indicates the permission status for the field when users submit entries. See `ARCreateField` information in BMC Remedy AR System C API functions for a description of the possible values. Specify `NULL` for this parameter if you do not want to retrieve this flag.

fieldOption

A bitmask that indicates whether the field is to be audited or copied when other fields are audited.

Bit 0:	Value 1: Audit this field. (`AR_FIELD_BITOPTION_AUDIT`)
Bit 1:	Value 1: Copy this field when other fields in the form are audited. (`AR_FIELD_BITOPTION_COPY`)
Bit 2:	Value 1: Indicates this field is for Log Key 1. (`AR_FIELD_BITOPTION_LOG_KEY1`)
Bit 3:	Value 1: Indicates this field is for Log Key 2. (`AR_FIELD_BITOPTION_LOG_KEY2`)
Bits 2 and 3:	Both value 1: Indicates this field is for Log Key 3. (`AR_FIELD_BITOPTION_LOG_KEY3`)

defaultValue

The value to apply if a user submits an entry with no field value (only applicable to data fields). The system returns `0` (`AR_DEFAULT_VALUE_NONE`) if the field has no default. Specify `NULL` for this parameter if you do not want to retrieve the default value.

permissionList

The list of group access permissions.

addedPermissionList

A set of groups that is added to the permission list when the permissions of an overlay component are additive. The list will be empty if an overlay is not retrieved or if the permissions of the overlay component are not of additive type.

Specify `NULL` for this parameter if you do not want to retrieve this value.

fieldLimit

The value limits for the field and other properties specific to the field's type. See Defining field limits for a description of the contained structure that applies to the type of field that you are creating. Specify `NULL` (or `AR_FIELD_LIMIT_NONE`) for trim and control fields or if you do not want to retrieve the limits and properties.

dInstanceList

A list of zero or more display properties associated with the field. See ARCreateField for a description of the possible values. The system returns `0` (`AR_DPROP_NONE`) if the field has no display properties. Specify `NULL` for this parameter if you do not want to retrieve this list.

owner

The owner for the field.

lastModifiedBy

The user who last modified the field.

modifiedDate

The date that the field was last modified.

helpText

The help text associated with the field. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the field.

arDocVersion

The XML document version.

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARGetFieldFromXML

10.4.34 ARSetSchemaWithExtendedFieldsToXML

Description

Updates the definition for the form with the indicated name on the specified server. If the schema is locked, only the indexList and the defaultVui can be set.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetSchemaWithExtendedFieldsToXML(
ARControlStruct         *control,
ARXMLOutputDoc          *xmlOutputDoc,
ARBoolean                xmlDocHdrFtrFlag,
ARNameType               schemaName,
ARCompoundSchema        *compoundSchema,
ARPermissionList        *permissionList,
ARInternalIdList        *subAdminGrpList,
AREntryListFieldList    *getListFields,
ARSortList              *sortList,
ARIndexList             *indexList,
ARArchiveInfoStruct     *archiveInfo,
ARAuditInfoStruct       *auditInfo,
ARNameType               defaultVUI,
ARInternalId            *nextFieldID,
ARULong32               *coreVersion,
int                     *upgradeVersion,
ARExportFieldInfoList   *fieldInfoList,
ARVuiInfoList           *vuiInfoList,
ARAccessNameType         owner,
ARAccessNameType         lastModifiedBy,
ARTimestamp             *modifiedDate,
char                    *helpText,
char                    *changeHistory,
ARPropList              *objPropList,
unsigned int            *arDocVersion,
ARStatusList            *status
)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

schemaName

The schema name. Each schema name must be unique.

compoundSchema

The definition of the candidate form or table that contains the candidate field to retrieve.

permissionList

A list of zero or more groups who can access the form. Access to this information is limited to users with BMC Remedy AR System administrator privileges only. Specify `NULL` for this parameter if you do not want to retrieve the group list.

subAdminGrpList

A list of zero or more groups who can administer this form (and the associated filters, escalations, and active links). Users must belong to both a specified group *and* the Subadministrator group to obtain these privileges. Specify `NULL` for this parameter if you do not want to retrieve the administrator group list.

getListField

A list of zero or more fields that identifies the default query list data for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve this value.

sortList

A list of zero or more fields that identifies the default sort order for retrieving form entries. Specify `NULL` for this parameter if you do not want to retrieve the default sort fields.

indexList

The set of zero or more indexes for the form. Specify `NULL` for this parameter if you do not want to retrieve the index list.

archiveInfo

If the form is to be archived, this is the archive information for the form. Specify `NULL` for this parameter if you do not want to create or set the archive information. These are the options for archive type:

`1:`	The form is selected to have entries copied and deleted to the archive form (`AR_ARCHIVE_FORM`).
`2:`	The form is selected to have entries deleted from the main form (`AR_ARCHIVE_DELETE`).
`32:`	The form is selected for archive but without attachment field (`AR_ARCHIVE_NO_ATTACHMENTS`).
`64:`	The form is selected for archive without diary fields (`AR_ARCHIVE_NO_DIARY`).

In addition to the archive type, `archiveInfo` also stores the form name (if archive type is `AR_ARCHIVE_FORM`), time to trigger the archive, archive qualification criteria, and name of the main form that is being archived.

auditInfo

If a form is to be audited, this is the audit information for the form. Specify `NULL` for this parameter if you do not want to create or set the audit information. These are the values for audit type:

`0`:	The selected form is not to be audited. (`AR_AUDIT_NONE`).
`1`:	Audit fields and copy fields on the selected form are copied to the audit shadow form (`AR_AUDIT_COPY`).
`2`:	Audit fields and copy fields on the selected form are copied to the audit log form (`AR_AUDIT_LOG`).

In addition to the audit type, `auditInfo` also stores the form name (if audit type is `AR_AUDIT_COPY` or `AR_AUDIT_LOG`) and the audit qualification criteria.

defaultVui

The label for the default view.

nextFieldID

The next default field identification number, which the server generates if users do not specify it.

coreVersion

The version number of the core fields.

upgradeVersion

The upgrade version for auto generated internal server schemas. To disable this option, set it to zero (`0`).

fieldInfoList

The list of schema fields.

vuiInfoList

The list of schema views.

owner

The schema owner.

lastModifiedBy

The user who last modified the schema.

modifiedDate

The date that the schema was last modified.

helpText

The help text associated with the schema. Specify `NULL` for this parameter if you do not want to retrieve the help text (which is useful if you are calling this function to verify whether an instance of this object exists).

changeHistory

The change history of the schema.

objPropList

The object properties of the schema.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetSchemaToXML

10.4.35 ARSetVUIWithExtendedFieldsToXML

Description

Updates the form view (VUI) with the indicated ID on the specified server.

Privileges

All users.

Synopsis

```
#include "arextern.h"

int ARSetVUIWithExtendedFieldsToXML(
ARControlStruct          *control,
ARXMLOutputDoc           *xmlOutputDoc,
ARBoolean                 xmlDocHdrFtrFlag,
ARNameType                schemaName,
ARVuiInfoList            *vuiInfoList,
ARExportFieldInfoList    *fieldInfoList,
ARTimestamp              *modifiedDate,
unsigned int            *arDocVersion,
ARStatusList            *status
)
```

Input arguments

control

The control record for the operation. It contains information about the user requesting the operation, where that operation is to be performed, and which session is used to perform it. The **user**, **sessionId**, and **server** fields are required.

xmlOutputDoc

The XML document output source.

xmlDocHdrFtrFlag

The header and footer flag for the XML document output source. If there is only one object in the XML document, set this value to (`0`).

schemaName

The name of the view. Each view name must be unique.

vuiInfoList

The list of schema views.

fieldInfoList

The list of view fields.

modifiedDate

The date that the view was last modified.

arDocVersion

The XML document version.

Return values

status

A list of zero or more notes, warnings, or errors generated from a call to this function. For a description of all possible values, see Error checking.

See also

ARSetVUIToXML

11 Retrieving entries from multiple forms

Use the following information to understand how to use the `ARGetListEntryWithMultiSchemaFields` function to perform dynamic joins, which retrieve entries from multiple forms at run time. The following topics are provided:

- About ARGetListEntryWithMultiSchemaFields
- Dynamic joins
- Recursive queries
- Value set queries
- Vendor form joins

11.1 About ARGetListEntryWithMultiSchemaFields

Use the `ARGetListEntryWithMultiSchemaFields` function in C or Java programs to query multiple forms at runtime.

The main operation performed by this function is a *dynamic join.* Unlike AR System join forms, which are created at design time, dynamic joins are performed at run time. They enable you to perform database queries across multiple forms-including view and vendor forms on an ad hoc basis. Use dynamic joins instead of AR System join forms in situations such as these:

- At design time, you cannot predict which forms need to be joined at run time. For example, use dynamic joins to enable users to create custom reports that pull data from multiple forms.
- Some of the forms that you need to join are dynamically created.
- You use a lot of join forms, and you want to reduce the time required to create them and the space that they take up in memory.

Note

For information about creating AR System join forms at design time, see Creating join forms.

Depending on the structure of your data, this function might also perform the following suboperations to support the dynamic join:

- *Recursive queries* — If a form contains hierarchical (parent-child) data, such as manager and employee relationships, this function can use *one* query to search the form recursively and retrieve the complete hierarchy and related attributes. Thus, to retrieve related records from such forms, you no longer need to issue hundreds of `ARGetListEntry` calls in recursive loops. See Recursive queries.
- *Value set queries* — This function can perform IN and NOT IN operations on a fixed value set and on a value set returned by a subquery. See Value set queries.

- *Refine query results by using aggregation functions, groupBy clauses and having clauses* — To refine the selection of data retrieved from the database and reduce the need for in-memory processing of large numbers of records, you can now apply aggregation functions along with **groupBy** and **having** clauses. (AR System releases prior to 7.6.02 do not support these options.)
- *Vendor form joins* — Joining vendor forms with other types of forms enables a single query to access data in both BMC Remedy AR System and external sources. See Using aggregate functions, groupBy clauses, and having clauses.

Note

Users cannot issue these queries from the mid tier because mid tier does not support the `ARGetListEntryWithMultiSchemaFields` function. You can create API programs, however, that use this function. The programs must be able to parse the user-specified query and provide appropriate parameters to the function.

Use the following information to know more about dynamic joins.

- Architecture of ARGetListEntryWithMultiSchemaFields
- Using aliases in dynamic joins
- Using access control fields in dynamic joins
- Using FTS with dynamic joins
- Understanding the results

11.1.1 Architecture of ARGetListEntryWithMultiSchemaFields

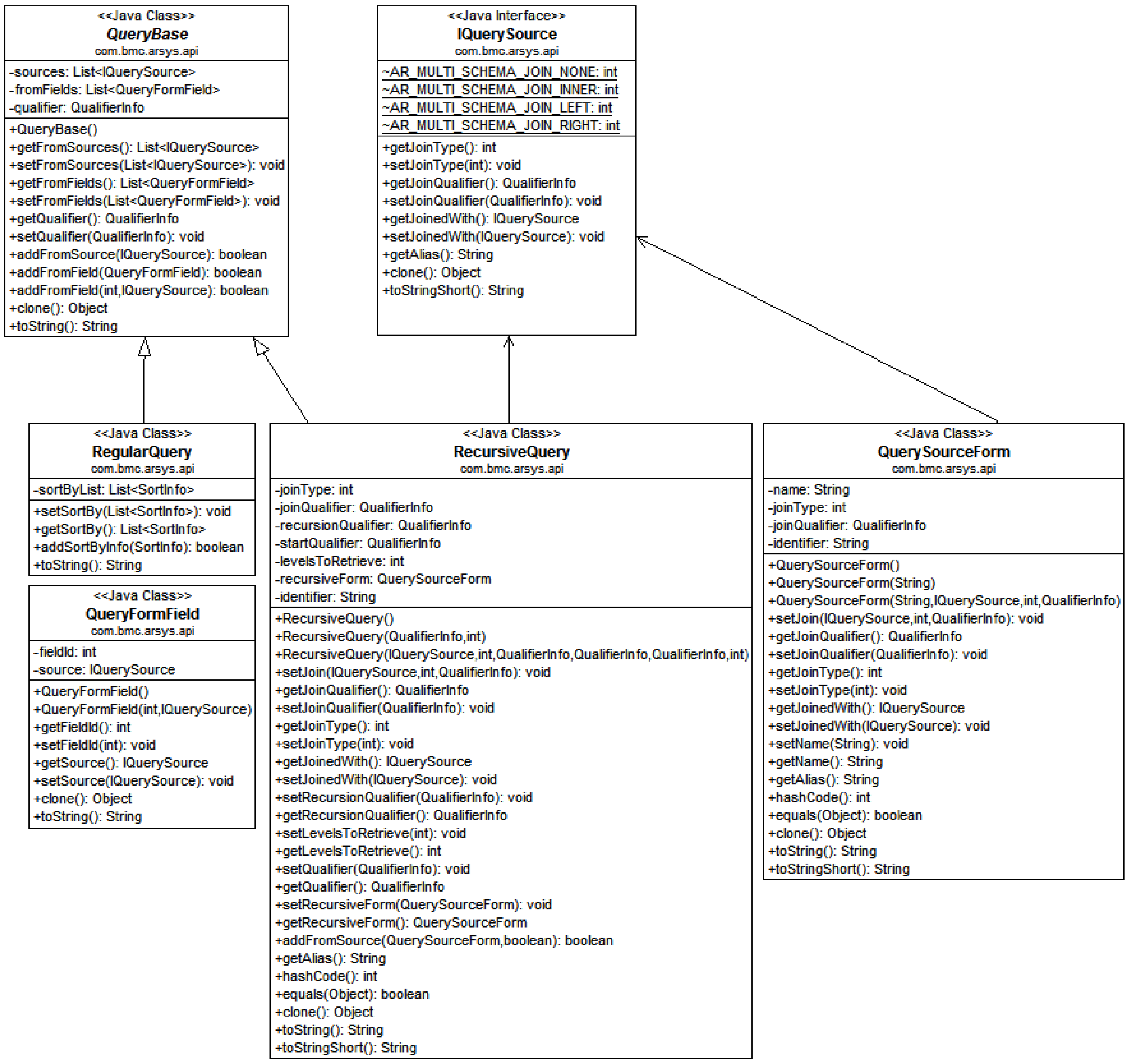
This section contains Java and C diagrams of the `ARGetListEntryWithMultiSchemaFields` function.

For detailed structural information, see these documents:

- (C API) ARGetListEntryWithMultiSchemaFields
- (C API) Structures for ARGetListEntryWithMultiSchemaFields
- (Java API) The Java API online documentation

Java class diagram

(Click the image to expand it.)

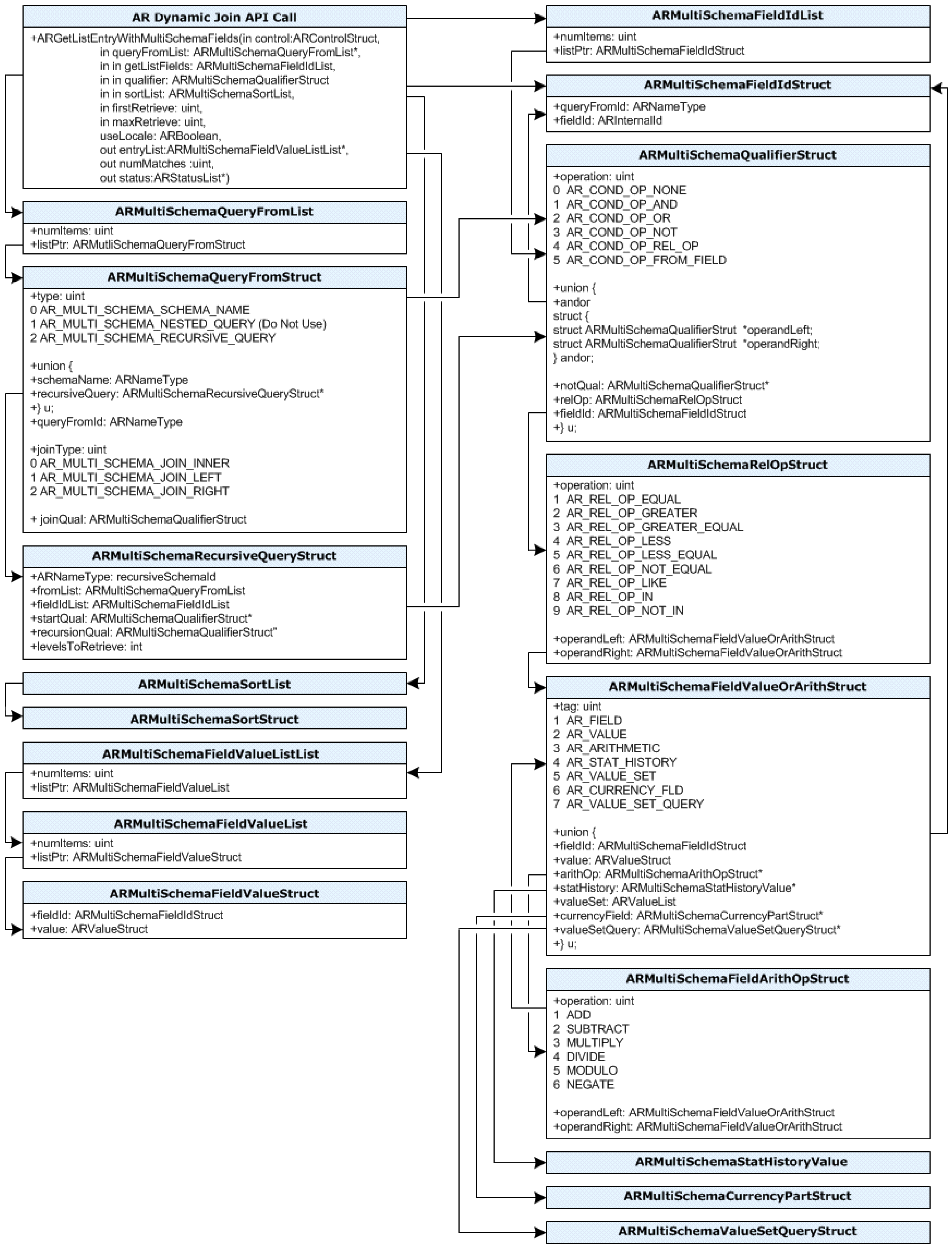

<<Java Class>>
QueryBase
com.bmc.arsys.api
-sources: List<IQuerySource>
-fromFields: List<QueryFormField>
-qualifier: QualifierInfo
+QueryBase()
+getFromSources(): List<IQuerySource>
+setFromSources(List<IQuerySource>): void
+getFromFields(): List<QueryFormField>
+setFromFields(List<QueryFormField>): void
+getQualifier(): QualifierInfo
+setQualifier(QualifierInfo): void
+addFromSource(IQuerySource): boolean
+addFromField(QueryFormField): boolean
+addFromField(int,IQuerySource): boolean
+clone(): Object
+toString(): String
<<Java Interface>>
IQuerySource
com.bmc.arsys.api
~AR_MULTI_SCHEMA_JOIN_NONE: int
~AR_MULTI_SCHEMA_JOIN_INNER: int
~AR_MULTI_SCHEMA_JOIN_LEFT: int
~AR_MULTI_SCHEMA_JOIN_RIGHT: int
+getJoinType(): int
+setJoinType(int): void
+getJoinQualifier(): QualifierInfo
+setJoinQualifier(QualifierInfo): void
+getJoinedWith(): IQuerySource
+setJoinedWith(IQuerySource): void
+getAlias(): String
+clone(): Object
+toStringShort(): String
<<Java Class>>
RegularQuery
com.bmc.arsys.api
-sortByList: List<SortInfo>
+setSortBy(List<SortInfo>): void
+getSortBy(): List<SortInfo>
+addSortByInfo(SortInfo): boolean
+toString(): String
<<Java Class>>
QueryFormField
com.bmc.arsys.api
-fieldId: int
-source: IQuerySource
+QueryFormField()
+QueryFormField(int,IQuerySource)
+getFieldId(): int
+setFieldId(int): void
+getSource(): IQuerySource
+setSource(IQuerySource): void
+clone(): Object
+toString(): String
<<Java Class>>
RecursiveQuery
com.bmc.arsys.api
-joinType: int
-joinQualifier: QualifierInfo
-recursionQualifier: QualifierInfo
-startQualifier: QualifierInfo
-levelsToRetrieve: int
-recursiveForm: QuerySourceForm
-identifier: String
+RecursiveQuery()
+RecursiveQuery(QualifierInfo,int)
+RecursiveQuery(IQuerySource,int,QualifierInfo,QualifierInfo,QualifierInfo,int)
+setJoin(IQuerySource,int,QualifierInfo): void
+getJoinQualifier(): QualifierInfo
+setJoinQualifier(QualifierInfo): void
+getJoinType(): int
+setJoinType(int): void
+getJoinedWith(): IQuerySource
+setJoinedWith(IQuerySource): void
+setRecursionQualifier(QualifierInfo): void
+getRecursionQualifier(): QualifierInfo
+setLevelsToRetrieve(int): void
+getLevelsToRetrieve(): int
+setQualifier(QualifierInfo): void
+getQualifier(): QualifierInfo
+setRecursiveForm(QuerySourceForm): void
+getRecursiveForm(): QuerySourceForm
+addFromSource(QuerySourceForm,boolean): boolean
+getAlias(): String
+hashCode(): int
+equals(Object): boolean
+clone(): Object
+toString(): String
+toStringShort(): String
<<Java Class>>
QuerySourceForm
com.bmc.arsys.api
-name: String
-joinType: int
-joinQualifier: QualifierInfo
-identifier: String
+QuerySourceForm()
+QuerySourceForm(String)
+QuerySourceForm(String,IQuerySource,int,QualifierInfo)
+setJoin(IQuerySource,int,QualifierInfo): void
+getJoinQualifier(): QualifierInfo
+setJoinQualifier(QualifierInfo): void
+getJoinType(): int
+setJoinType(int): void
+getJoinedWith(): IQuerySource
+setJoinedWith(IQuerySource): void
+setName(String): void
+getName(): String
+getAlias(): String
+hashCode(): int
+equals(Object): boolean
+clone(): Object
+toString(): String
+toStringShort(): String

Note

The Java class diagram and C structure diagram do not reflect the changes introduced in release 7.6.03, which allow you to use aggregate functions, group by clauses, and having clauses.

C structure diagram

(Click the image to expand it.)


AR Dynamic Join API Call
+ARGetListEntryWithMultiSchemaFields(in control:ARControlStruct,
in queryFromList: ARMultiSchemaQueryFromList*,
in in getListFields: ARMultiSchemaFieldIdList,
in in qualifier: ARMultiSchemaQualifierStruct
in in sortList: ARMultiSchemaSortList,
in firstRetrieve: uint,
in maxRetrieve: uint,
useLocale: ARBoolean,
out entryList:ARMultiSchemaFieldValueListList*,
out numMatches :uint,
out status:ARStatusList*)
ARMultiSchemaQueryFromList
+numItems: uint
+listPtr: ARMutliSchemaQueryFromStruct
ARMultiSchemaQueryFromStruct
+type: uint
0 AR_MULTI_SCHEMA_SCHEMA_NAME
1 AR_MULTI_SCHEMA_NESTED_QUERY (Do Not Use)
2 AR_MULTI_SCHEMA_RECURSIVE_QUERY
+union {
+schemaName: ARNameType
+recursiveQuery: ARMultiSchemaRecursiveQueryStruct*
+} u;
+queryFromId: ARNameType
+joinType: uint
0 AR_MULTI_SCHEMA_JOIN_INNER
1 AR_MULTI_SCHEMA_JOIN_LEFT
2 AR_MULTI_SCHEMA_JOIN_RIGHT
+ joinQual: ARMultiSchemaQualifierStruct
ARMultiSchemaRecursiveQueryStruct
+ARNameType: recursiveSchemaId
+fromList: ARMultiSchemaQueryFromList
+fieldIdList: ARMultiSchemaFieldIdList
+startQual: ARMultiSchemaQualifierStruct*
+recursionQual: ARMultiSchemaQualifierStruct"
+levelsToRetrieve: int
ARMultiSchemaSortList
ARMultiSchemaSortStruct
ARMultiSchemaFieldValueListList
+numItems: uint
+listPtr: ARMultiSchemaFieldValueList
ARMultiSchemaFieldValueList
+numItems: uint
+listPtr: ARMultiSchemaFieldValueStruct
ARMultiSchemaFieldValueStruct
+fieldId: ARMultiSchemaFieldIdStruct
+value: ARValueStruct
ARMultiSchemaFieldIdList
+numItems: uint
+listPtr: ARMultiSchemaFieldIdStruct
ARMultiSchemaFieldIdStruct
+queryFromId: ARNameType
+fieldId: ARInternalId
ARMultiSchemaQualifierStruct
+operation: uint
0 AR_COND_OP_NONE
1 AR_COND_OP_AND
2 AR_COND_OP_OR
3 AR_COND_OP_NOT
4 AR_COND_OP_REL_OP
5 AR_COND_OP_FROM_FIELD
+union {
+andor
struct {
struct ARMultiSchemaQualifierStrut *operandLeft;
struct ARMultiSchemaQualifierStrut *operandRight;
} andor;
+notQual: ARMultiSchemaQualifierStruct*
+relOp: ARMultiSchemaRelOpStruct
+fieldId: ARMultiSchemaFieldIdStruct
+} u;
ARMultiSchemaRelOpStruct
+operation: uint
1 AR_REL_OP_EQUAL
2 AR_REL_OP_GREATER
3 AR_REL_OP_GREATER_EQUAL
4 AR_REL_OP_LESS
5 AR_REL_OP_LESS_EQUAL
6 AR_REL_OP_NOT_EQUAL
7 AR_REL_OP_LIKE
8 AR_REL_OP_IN
9 AR_REL_OP_NOT_IN
+operandLeft: ARMultiSchemaFieldValueOrArithStruct
+operandRight: ARMultiSchemaFieldValueOrArithStruct
ARMultiSchemaFieldValueOrArithStruct
+tag: uint
1 AR_FIELD
2 AR_VALUE
3 AR_ARITHMETIC
4 AR_STAT_HISTORY
5 AR_VALUE_SET
6 AR_CURRENCY_FLD
7 AR_VALUE_SET_QUERY
+union {
+fieldId: ARMultiSchemaFieldIdStruct
+value: ARValueStruct
+arithOp: ARMultiSchemaArithOpStruct*
+statHistory: ARMultiSchemaStatHistoryValue*
+valueSet: ARValueList
+currencyField: ARMultiSchemaCurrencyPartStruct*
+valueSetQuery: ARMultiSchemaValueSetQueryStruct*
+} u;
ARMultiSchemaFieldArithOpStruct
+operation: uint
1 ADD
2 SUBTRACT
3 MULTIPLY
4 DIVIDE
5 MODULO
6 NEGATE
+operandLeft: ARMultiSchemaFieldValueOrArithStruct
+operandRight: ARMultiSchemaFieldValueOrArithStruct
ARMultiSchemaStatHistoryValue
ARMultiSchemaCurrencyPartStruct
ARMultiSchemaValueSetQueryStruct

11.1.2 Using aliases in dynamic joins

In the BMC Remedy AR System C API, the `ARMultiSchemaFuncQueryFromStruct` structure (see ARMultiSchemaRecursiveFuncQueryStruct) of the `ARGetListEntryWithMultiSchemaFields` function represents an item. An item can be a form (schema) or a recursive query. This structure uses a union to support recursive queries. See ARMultiSchemaFuncQueryFromStruct.

One of the elements of this structure is the item's alias (queryFromAlias). An alias is required for recursive queries (see ARMultiSchemaRecursiveFuncQueryStruct) and for forms involved in self joins. (In self joins, the same form appears more than once in the `queryFromList` parameter, so each occurrence of the form requires a unique alias.)

In C, when the function asks for a reference, you must give it a field ID *and* an alias.

In the BMC Remedy AR System Java API, the `queryFromAlias` is not exposed. Instead, the function uses GUIDs to generate aliases automatically. Hence, wherever you would specify an alias in C, just specify a reference to the `IQuerySource` object in Java.

11.1.3 Using access control fields in dynamic joins

Although the following operations are performed on the same forms, they might return different results if the forms contain the Assignee Group field:

- Searching an AR System join form composed of Form A and Form B
- Issuing an `ARGetListEntryWithMultiSchemaFields` call across Form A and Form B

If both underlying forms in an AR System join form contain the Assignee Group field (field ID 112), the join form inherits row-level access permissions from only the primary form. In queries across multiple forms, however, row-level access permissions for each form referenced in the query are evaluated separately; one form does not control access to the other.

See also Access control fields and value set queries.

11.1.4 Using FTS with dynamic joins

Full text search (FTS) works the same in dynamic joins as it does elsewhere with one exception: `ARGetListEntryWithMultiSchemaFields` ignores FTS weights when sorting the records it retrieves. Therefore, do not include the **WEIGHT** field (field ID 99) in the function's `sortList` parameter.

11.1.5 Understanding the results

The format of the `ARGetListEntryWithMultiSchemaFields` results differs from the format of the `ARGetListEntryWithFields` results as follows:

Function	Records returned by function include	

	Entry ID	Field/value list
`ARGetListEntryWithFields`	Yes	Yes
`ARGetListEntryWithMultiSchemaFields`	No	Yes

To include request IDs in `ARGetListEntryWithMultiSchemaFields` results, you must include the Request ID field in the function's `getListFields` parameter.

11.2 Dynamic joins

The following C and Java examples show how to join two forms, Printer and Computer, dynamically.

The forms are joined by matching the values in the Printer ID and Computer ID fields (the join qualifier).

- C example--dynamic joins
- Java example--dynamic joins
- SQL generated by the call--dynamic joins

11.2.1 C example--dynamic joins

This example is updated to show where existing code would need to change to use the **groupBy** and **having** clauses and related structures introduced in release 7.6.03. Lines with required code changes are marked with ***//***.

For information about using the **groupBy** and **having** clauses and the related aggregate functions, see Using aggregate functions, groupBy clauses, and having clauses.

```
void queryPrinterAndComputer(ARControlStruct *ctrl, ARStatusList *status)
{
   ARMultiSchemaFieldFuncValueListList msEntryList = {0, NULL}; /**/
   ARMultiSchemaFuncQueryFromList fromList = {0, NULL}; /**/
   ARMultiSchemaFieldFuncList getListFields = {0, NULL}; /**/
   ARMultiSchemaSortList sortList;
   unsigned int numMatches;
   unsigned int *numMatchesPtr;
   int result;
   ARMultiSchemaFuncQueryFromStruct fromItem[2] = {{0,},}; /* 0-fill */ /**/
   ARMultiSchemaQualifierStruct *qualifier;
   ARMultiSchemaQualifierStruct wherequal={0, NULL};
   ARMultiSchemaQualifierStruct *qual=NULL;
   ARMultiSchemaRelOpStruct relop;
   ARMultiSchemaFieldFuncStruct fieldId[7] = {{"",},}; /* 0-fill */ /**/
   ARMultiSchemaFieldFuncStruct fieldId[7] = {{"",},}; /* 0-fill */ /**/
   /* fill up query struct */
   fromList.numItems = 2;
   fromList.listPtr = fromItem;
   qualifier = &wherequal; /*empty qual*/
   /* The following sort list is used to generate the SQL query's ORDER BY */
   /* clause. It instructs the function to sort first on the Printer ID field */
   /* and then on the Computer ID field. */
   sortList.numItems = 2;
   sortList.listPtr = (ARMultiSchemaSortStruct*)
      malloc(sizeof(ARMultiSchemaSortStruct) * sortList.numItems);
   sortList.listPtr[0].fieldId.fieldId = 536870914; // Printer ID
   strcpy(sortList.listPtr[0].fieldId.queryFromAlias, "Printer");
   sortList.listPtr[0].sortOrder = 1;
   sortList.listPtr[1].fieldId.fieldId = 536870914; // Computer ID
   strcpy(sortList.listPtr[1].fieldId.queryFromAlias, "Computer");
   sortList.listPtr[1].sortOrder = 1;
   fromItem[0].type = AR_MULTI_SCHEMA_SCHEMA_NAME;
   strcpy(fromItem[0].u.schemaName, "Printer");
   fromItem[0].joinType = 0;
   fromItem[0].joinQual = NULL;
   fromItem[1].type = AR_MULTI_SCHEMA_SCHEMA_NAME;
   strcpy(fromItem[1].u.schemaName, "Computer");
   fromItem[1].joinType = 0;
   fromItem[1].joinQual = qualifier;
   numMatchesPtr = &numMatches;
   /* The following fields are used in the getListFields parameter, which */
   /* specifies the fields to include in the results. This parameter */
   /* generates the SQL query's SELECT list. */
   strcpy(fieldId[0].queryFromAlias, "Printer");
   fieldId[0].fieldId = 1; /*Printer entryId */
   fieldId[0].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   strcpy(fieldId[1].queryFromAlias, "Printer");
   fieldId[1].fieldId = 536870913; /* Printer name */
   fieldId[1].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   strcpy(fieldId[2].queryFromAlias, "Printer");
   fieldId[2].fieldId = 536870914; /* Printer ID */
   fieldId[2].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   strcpy(fieldId[3].queryFromAlias, "Printer");
   fieldId[3].fieldId = 536870915; /* Printer type */
   fieldId[3].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   strcpy(fieldId[4].queryFromAlias, "Computer");
   fieldId[4].fieldId = 1; /* Computer entryId */
   fieldId[4].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   strcpy(fieldId[5].queryFromAlias, "Computer");
   fieldId[5].fieldId = 536870913; /* Computer name */
   fieldId[5].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   strcpy(fieldId[6].queryFromAlias, "Computer");
   fieldId[6].fieldId = 536870914; /* Computer ID */
   fieldId[6].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   getListFields.numItems = 7;
   getListFields.listPtr = fieldId;
   /* This is the join qualifier, which is used to generate the SQL query's */
   /* ON clause. */
   qualifier->operation = AR_COND_OP_REL_OP;
   qualifier->u.relOp = &relop;
   relop.operation = AR_REL_OP_EQUAL;
   relop.operandLeft.tag = AR_FIELD;
   strcpy(relop.operandLeft.u.fieldId.queryFromAlias, "Printer");
   relop.operandLeft.u.fieldId.fieldId = 536870914;
   relop.operandRight.tag = AR_FIELD;
   strcpy(relop.operandRight.u.fieldId.queryFromAlias, "Computer");
   relop.operandRight.u.fieldId.fieldId = 536870914;
   /* Call routine */
   result = ARGetListEntryWithMultiSchemaFields(ctrl, &fromList,
         &getListFields, qualifier, &sortList, AR_START_WITH_FIRST_ENTRY,
         AR_NO_MAX_LIST_RETRIEVE, FALSE,
         NULL, NULL,  /* groupBy and having */ /**/
         &msEntryList, numMatchesPtr, status);
}
```

11.2.2 Java example--dynamic joins

```
private void queryPrinterAndComputer(ARServerUser arConnection) throws ARException
{
    RegularQuery query = new RegularQuery();

    // Add Printer and Computer forms as sources.
    QuerySourceForm printerForm = new QuerySourceForm("Printer");
    QuerySourceForm computerForm = new QuerySourceForm("Computer");
    query.addFromSource(printerForm);
    query.addFromSource(computerForm);

    // Create join qualifier, which is used to generate the SQL query's
    // ON clause.
    ArithmeticOrRelationalOperand field1 =
    new ArithmeticOrRelationalOperand(536870914, printerForm);
    ArithmeticOrRelationalOperand field2 =
    new ArithmeticOrRelationalOperand(536870914, computerForm);
    RelationalOperationInfo relOp =
    new RelationalOperationInfo(RelationalOperationInfo.AR_REL_OP_EQUAL,
    field1, field2);
    QualifierInfo joinQual = new QualifierInfo(relOp);
    computerForm.setJoin(printerForm, IQuerySource.AR_MULTI_SCHEMA_JOIN_INNER,
    joinQual);

    // The following fields are included in the results. This parameter
    // generates the SQL query's SELECT list.
    query.addFromField(1, printerForm);          // Printer entry ID
    query.addFromField(536870913, printerForm);  // Printer name
    query.addFromField(536870914, printerForm);  // Printer ID
    query.addFromField(536870915, printerForm);  // Printer type

    query.addFromField(1, computerForm);          // Computer entry ID
    query.addFromField(536870913, computerForm);  // Computer name
    query.addFromField(536870914, computerForm);  // Computer ID

    // The following SortInfos are used to generate the SQL query's ORDER BY
    // clause. It specifies to sort first on the Printer ID field and then on
    // the Computer ID field.
    query.addSortByInfo(new SortInfo(536870914, printerForm,
    Constants.AR_SORT_ASCENDING));
    query.addSortByInfo(new SortInfo(536870914, computerForm,
    Constants.AR_SORT_ASCENDING));

    // Call routine.
    List<QuerySourceValues> results =
    arConnection.getListEntryObjects(query, 0, 0, false, null);
}
```

11.2.3 SQL generated by the call--dynamic joins

This section shows the SQL generated by the preceding examples.

Note

In this example, an AR System administrator made the API call, and the form did not use row-level security. If a nonadministrator makes the API call and the form uses row-level security, the SQL generated by the call will differ from this example.

```
SELECT J0.C1, J0.C536870913, J0.C536870914, J0.C536870915,
       J1.C1, J1.C536870913, J1.C536870914
FROM T169 J0 INNER JOIN T168 J1 ON (J0.C536870914 = J1.C536870914)
ORDER BY J0.C536870914 ASC, J1.C536870914 ASC, J0.C1 ASC
```

11.3 Recursive queries

Recursive queries retrieve information from forms that store hierarchical (parent-child) data. For example, a Bill of Materials form might contain information about the relationship between assemblies, their subassemblies, the subassemblies' subassemblies, and so on.

The ARMultiSchemaFuncQueryFromList structure of the ARGetListEntryWithMultiSchemaFields function represents a list of items that are included in the **FROM** clause of the query generated by the function. The items can include multiple forms (schemas) and *one* recursive query.

The ARMultiSchemaRecursiveFuncQueryStruct structure represents the recursive query. You can return hierarchical data from only one form. Specify that form in the structure's **queryFromList** parameter, and then give it an alias (**recursiveSchemaAlias**). To specify the hierarchical node or nodes to begin the recursion on, use the start with qualifier (***startQual**). To apply conditions from other forms during the recursion, add those forms to the recursion qualifier (***recursionQual**).

For example, to retrieve a list of all employees who work for Jane Smith in the Seattle office, set up the appropriate recursive query on the Employee form, which contains hierarchical information about relationships between employees and managers. In the **levelsToRetrieve** element, specify 0 to search all levels in the hierarchy. In the recursion qualifier, specify that the value in the City field on the Location form for each employee must be Seattle. The recursive query finds 35 employees who work for Jane: two reporting directly and 33 reporting either to her direct reports or to subordinates of her direct reports. Only 12 of those employees, however, work in the Seattle office, so the query returns only 12 records.

The following C and Java examples show how to use a recursive query to find all employees who either report directly to Bob Jones or who report to his direct reports. The query retrieves the root or parent level (**startQual**) and two levels below the root.

Note

For recursive queries, set the sortList parameter to NULL.

- C example--recursive queries
- Java example --recursive queries
- SQL generated by the query--recursive queries

11.3.1 C example--recursive queries

This example is updated to show where existing code would need to change to use the **groupBy** and **having** clauses and related structures introduced in release 7.6.03. Lines with required code changes are marked with ***//***.

For information about using the **groupBy** and **having** clauses and the related aggregate functions, see Using aggregate functions, groupBy clauses, and having clauses.

```
#define EMPLOYEE_FORM "Employee"
#define EMPLOYEE_ALIAS "Employee"
#define EMP_EMPLOYEE_NAME 536870913
#define EMP_EMPLOYEE_ID 536870914
#define EMP_MANAGER_ID 536870915
#define EMP_MANAGER_ID 536870915
void queryOnEmployeeForm(ARControlStruct *ctrl, ARStatusList *status)
{
   ARMultiSchemaFieldFuncValueListList msEntryList = {0, NULL};  /**/
   unsigned int numMatches;
   unsigned int *numMatchesPtr;
   int result;
   ARMultiSchemaFuncQueryFromList fromList = {0, NULL};  /**/
   ARMultiSchemaSortList sortList = {0, NULL};
   ARMultiSchemaFieldFuncList getListFields = {0, NULL};  /**/
   ARMultiSchemaFuncQueryFromStruct fromItem = {0,}; /* 0-fill*/ /**/
   ARMultiSchemaRecursiveFuncQueryStruct recursiveQuery = {"",}; /* 0-fill */ /**/
   ARMultiSchemaFuncQueryFromStruct recqueryfromItem = {0,};  /* 0-fill */ /**/
   ARMultiSchemaQualifierStruct qual = {AR_COND_OP_NONE, NULL};
   ARMultiSchemaQualifierStruct newQual = {AR_COND_OP_NONE, NULL};
   ARMultiSchemaQualifierStruct qualifier = {AR_COND_OP_NONE, NULL};
   ARMultiSchemaQualifierStruct startWith = {AR_COND_OP_NONE, NULL};
   ARMultiSchemaQualifierStruct connectBy = {AR_COND_OP_NONE, NULL};
   ARMultiSchemaRelOpStruct swRelop, relop;
   ARMultiSchemaRelOpStruct cbRelop;
   ARMultiSchemaFieldFuncStruct fieldIdRecQuery[4] = {{"",},}; /* 0-fill */ /**/
   ARMultiSchemaFieldFuncStruct fieldIdMain[4] = {{"",},}; /* 0-fill*/ /**/
   /* fill up query struct */
   fromList.numItems = 1;
   fromList.listPtr = &fromItem;
   strcpy(fieldIdMain[0].queryFromAlias, "recquery");
   fieldIdMain[0].fieldId = EMP_EMPLOYEE_NAME; /* employee name */
   fieldIdMain[0].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
```

```
strcpy(fieldIdMain[1].queryFromAlias, "recquery");
fieldIdMain[1].fieldId = EMP_EMPLOYEE_ID; /* employee id */
fieldIdMain[1].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
strcpy(fieldIdMain[2].queryFromAlias, "recquery");
fieldIdMain[2].fieldId = EMP_MANAGER_ID; /* manager id */
fieldIdMain[2].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
getListFields.numItems = 3;
getListFields.listPtr = fieldIdMain;  /**/
fromItem.type = AR_MULTI_SCHEMA_RECURSIVE_QUERY;
fromItem.u.recursiveQuery = &recursiveQuery;
fromItem.joinType = 0;
fromItem.joinQual = NULL;
strcpy(fromItem.queryFromAlias, "recquery");
/* The following lines specify the form on which the recursive query */
/* operates. They also specify a recursive schema alias for that form. */
recursiveQuery.queryFromList.numItems = 1;
recursiveQuery.queryFromList.listPtr = &recqueryfromItem;
recursiveQuery.getListFuncs.numItems = 3;
strcpy(fieldIdRecQuery[0].queryFromAlias, EMPLOYEE_ALIAS);
fieldIdRecQuery[0].fieldId = EMP_EMPLOYEE_NAME; /* employee name */
fieldIdRecQuery[0].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
strcpy(fieldIdRecQuery[1].queryFromAlias, EMPLOYEE_ALIAS);
fieldIdRecQuery[1].fieldId = EMP_EMPLOYEE_ID; /* employee id */
fieldIdRecQuery[1].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
strcpy(fieldIdRecQuery[2].queryFromAlias, EMPLOYEE_ALIAS);
fieldIdRecQuery[2].fieldId = EMP_MANAGER_ID; /* manager id */
fieldIdRecQuery[2].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
recursiveQuery.getListFuncs.listPtr = fieldIdRecQuery; /**/
recursiveQuery.startQual = &startWith;
recursiveQuery.recursionQual = &connectBy;
/* The following line specifies the number of levels in the hierarchy to */
/* retrieve. Specify 3 to retrieve 2 levels below the parent level. (The */
/* parent level is identified by the startQual parameter.) */
recursiveQuery.levelsToRetrieve = 3;
strcpy(recursiveQuery.recursiveSchemaAlias, EMPLOYEE_ALIAS);
recqueryfromItem.type = AR_MULTI_SCHEMA_SCHEMA_NAME;
strcpy(recqueryfromItem.u.schemaName, EMPLOYEE_FORM);
recqueryfromItem.joinType = 0;
recqueryfromItem.joinQual = NULL;
strcpy(recqueryfromItem.queryFromAlias, EMPLOYEE_ALIAS);
startWith.operation = AR_COND_OP_REL_OP;
startWith.u.relOp = &swRelop;
swRelop.operation = AR_REL_OP_EQUAL;
swRelop.operandLeft.tag = AR_FIELD;
strcpy(swRelop.operandLeft.u.fieldId.queryFromAlias, EMPLOYEE_ALIAS);
swRelop.operandLeft.u.fieldId.fieldId = EMP_EMPLOYEE_NAME;
swRelop.operandRight.tag = AR_VALUE;
swRelop.operandRight.u.value.dataType = AR_DATA_TYPE_CHAR;
swRelop.operandRight.u.value.u.charVal = "Bob Jones";
/* The following lines specify the recursion qualifier. */
connectBy.operation = AR_COND_OP_REL_OP;
connectBy.u.relOp = &cbRelop;
cbRelop.operation = AR_REL_OP_EQUAL;
cbRelop.operandLeft.tag = AR_FIELD;
strcpy(cbRelop.operandLeft.u.fieldId.queryFromAlias, EMPLOYEE_ALIAS); /* child */
cbRelop.operandLeft.u.fieldId.fieldId = EMP_MANAGER_ID; /* manager id */
cbRelop.operandRight.tag = AR_FIELD;
strcpy(cbRelop.operandRight.u.fieldId.queryFromAlias,
         AR_REC_QRY_PARENT_ALIAS); /* parent */
   cbRelop.operandRight.u.fieldId.fieldId = EMP_EMPLOYEE_ID; /* employee id */
   numMatchesPtr = &numMatches;
   /* Call routine */
   result = ARGetListEntryWithMultiSchemaFields (ctrl, &fromList,
         &getListFields, &qualifier, &sortList, AR_START_WITH_FIRST_ENTRY,
         AR_NO_MAX_LIST_RETRIEVE, FALSE,
         NULL, NULL, /* groupBy, having */ /**/
         &msEntryList, numMatchesPtr, status);
}
```

11.3.2 Java example --recursive queries

```
private static final String EMPLOYEE_FORM = "Employee";
private static final int EMP_EMPLOYEE_NAME = 536870913;
private static final int EMP_EMPLOYEE_ID = 536870914;
private static final int EMP_MANAGER_ID = 536870915;
private static final int EMP_MANAGER_ID = 536870915;
private void queryOnEmployeeForm(ARServerUser arConnection) throws ARException
{
    // Create recursive query part first.
    RecursiveQuery recursive = new RecursiveQuery();

    // Add Employee form as the source. Since there is only
    // one source you do not need to specify recursive source with
    // the setRecursiveForm method.
    QuerySourceForm employeeForm = new QuerySourceForm(EMPLOYEE_FORM);
    recursive.addFromSource(employeeForm);

    // The following fields are included in the results of the recursion.
    recursive.addFromField(EMP_EMPLOYEE_NAME, employeeForm);
    recursive.addFromField(EMP_EMPLOYEE_ID, employeeForm);
    recursive.addFromField(EMP_MANAGER_ID, employeeForm);

    // Create the starting qualifier.
    ArithmeticOrRelationalOperand startFieldOp =
    new ArithmeticOrRelationalOperand(EMP_EMPLOYEE_NAME, employeeForm);
    ArithmeticOrRelationalOperand startValueOp =
    new ArithmeticOrRelationalOperand(new Value("Bob Jones"));
    RelationalOperationInfo startRelOp =
    new RelationalOperationInfo(RelationalOperationInfo.AR_REL_OP_EQUAL,
    startFieldOp, startValueOp);
    QualifierInfo startQual = new QualifierInfo(startRelOp);
    recursive.setQualifier(startQual);

    // Create the recursion qualifier.
    ArithmeticOrRelationalOperand recurseField1 =
    new ArithmeticOrRelationalOperand(EMP_MANAGER_ID, employeeForm);
    ArithmeticOrRelationalOperand recurseField2 =
    new ArithmeticOrRelationalOperand(EMP_EMPLOYEE_ID, recursive);
    RelationalOperationInfo recurseRelOp =
    new RelationalOperationInfo(RelationalOperationInfo.AR_REL_OP_EQUAL,
    recurseField1, recurseField2);
    QualifierInfo recurseQual = new QualifierInfo(recurseRelOp);
    recursive.setRecursionQualifier(recurseQual);

    // Specify the number of levels in the hierarchy to retrieve.
    // Specify 3 to retrieve 2 levels below the parent level.
    // (The parent level is identified by the starting qualifier.)
    recursive.setLevelsToRetrieve(3);

    // Create regular query.
    RegularQuery query = new RegularQuery();

    // Add recursive query as source.
    query.addFromSource(recursive);

    // Add the fields from the recursive query as the return field
    // in the regular query.
    query.addFromField(EMP_EMPLOYEE_NAME, recursive);
    query.addFromField(EMP_EMPLOYEE_ID, recursive);
    query.addFromField(EMP_MANAGER_ID, recursive);

    // Call routine.
    List<QuerySourceValues> results =
    arConnection.getListEntryObjects(query, 0, 0, false, null);
}
```

11.3.3 SQL generated by the query--recursive queries

This section shows the SQL generated by the preceding recursive query. Different databases generate different SQL.

Note

In these examples, an AR System administrator made the API call, and the form did not use row-level security. If a nonadministrator makes the API call and the form uses row-level security, the SQL generated by the call will differ from these examples.

IBM DB2 and Microsoft SQL Server

The SQL generated by DB2 and Microsoft SQL Server databases uses the Common Table Expressions (CTE) implementation:

```
; WITH T170_3244 (C536870913, C536870914, C536870915, LEVEL) AS
(
    SELECT J0.C536870913, J0.C536870914, J0.C536870915, 1
    FROM T170 J0
    WHERE (J0.C536870913 = N'Bob Jones')
    UNION ALL
    SELECT J0.C536870913, J0.C536870914, J0.C536870915, J1.LEVEL+1
    FROM T170 J0, T170_3244 J1
    WHERE (J0.C536870915 = J1.C536870914) AND (J1.LEVEL < 3)
)
```

```
SELECT J0.C536870913, J0.C536870914, J0.C536870915
FROM T170_3244 J0
```

Oracle

The SQL generated by Oracle databases uses the native Oracle implementation:

```
SELECT J0.C536870913 a0, J0.C536870914 a1, J0.C536870915 a2 FROM
(
    SELECT J0.C536870913, J0.C536870914, J0.C536870915
    FROM T2672 J0
    START WITH (J0.C536870913 = 'Bob Jones')
    CONNECT BY (J0.C536870915 = PRIOR J0.C536870914 AND LEVEL <= 3)
) J0
```

Sybase ASE

The SQL generated by Sybase databases uses temporary tables created by BMC Remedy AR System:

```
SELECT J0.C536870913, J0.C536870914, J0.C536870915, 1 RECLEVEL
INTO #T2670_4390
FROM T2670 J0
WHERE (J0.C536870913 = 'Bob Jones')
WHERE (J0.C536870913 = 'Bob Jones')
INSERT INTO #T2670_4390
SELECT J0.C536870913, J0.C536870914, J0.C536870915, 2
FROM T2670 J0, #T2670_4390 J1
WHERE (J0.C536870915 = J1.C536870914) AND J1.RECLEVEL = 1
WHERE (J0.C536870915 = J1.C536870914) AND J1.RECLEVEL = 1
INSERT INTO #T2670_4390
SELECT J0.C536870913, J0.C536870914, J0.C536870915, 3
FROM T2670 J0, #T2670_4390 J1
WHERE (J0.C536870915 = J1.C536870914) AND J1.RECLEVEL = 2
WHERE (J0.C536870915 = J1.C536870914) AND J1.RECLEVEL = 2
SELECT J0.C536870913, J0.C536870914, J0.C536870915 FROM #T2670_4390 J0
SELECT J0.C536870913, J0.C536870914, J0.C536870915 FROM #T2670_4390 J0
DROP TABLE #T2670_4390
```

Note

When a recursive query contains outer joins, a temporary table is used to hold the results of the recursive query in *all* databases.

11.4 Value set queries

The ARGetListEntryWithMultiSchemaFields function can perform IN and NOT IN operations on a fixed value set and on a value set returned by a subquery (the **WHERE** clause of an SQL query).

The function's ARMultiSchemaFuncValueSetQueryStruct structure represents a subquery for the IN and NOT IN operators. Each IN and NOT IN operation can be performed on only one field.

The following C and Java examples show a value set query. In these examples, the login user name is InSubqueryUser. The query operates on the **InSubqueryPermissions1** form and performs the IN subquery on the **InSubqueryPermissions3** form. The example assumes that both forms are populated with the appropriate data. The following topics are provided:

- C example--value set queries
- Java example--value set queries
- SQL generated by the call--value set queries
- Access control fields and value set queries
- Using aggregate functions, groupBy clauses, and having clauses

11.4.1 C example--value set queries

This example is updated to show where existing code would need to change to use the **groupBy** and **having** clauses and related structures introduced in release 7.6.03. Lines with required code changes are marked with ***//***.

For information about using the **groupBy** and **having** clauses and the related aggregate functions, see Using aggregate functions, groupBy clauses, and having clauses.

```
void queryUsingSubquery(unsigned int IN_NOT_IN_op,
   ARControlStruct *ctrl, ARStatusList *status)
{
   ARMultiSchemaFieldFuncValueListList msEntryList = {0, NULL};  /**/
   unsigned int numMatches;
   unsigned int *numMatchesPtr;
   int result;
   ARMultiSchemaFuncQueryFromList fromList = {0, NULL};  /**/
   ARMultiSchemaSortList sortList = {0, NULL};
   ARMultiSchemaFieldFuncList getListFields = {0, NULL};  /**/
   ARMultiSchemaFuncQueryFromStruct perm1fromItem = {0,}; /* 0-fill */ /**/
   ARMultiSchemaFuncQueryFromStruct perm3fromItem = {0,}; /* 0-fill */ /**/
   ARMultiSchemaQualifierStruct joinQual = {AR_COND_OP_NONE, NULL};
   ARMultiSchemaQualifierStruct qualifier = {AR_COND_OP_NONE, NULL};
   ARMultiSchemaRelOpStruct relop;
   ARMultiSchemaFieldFuncStruct fieldIdMain[4] = {{"",}};  /* 0-fill */ /**/
   ARMultiSchemaValueSetFuncQueryStruct valueSetQuery = {{0,}}; /* 0-fill *//**/
   /* fill up query struct */
   fromList.numItems = 1;
   fromList.listPtr = &perm1fromItem;
   strcpy(fieldIdMain[0].queryFromAlias, "perm1");
   fieldIdMain[0].fieldId = 1; /*requestId */
   fieldIdMain[0].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   strcpy(fieldIdMain[1].queryFromAlias, "perm1");
   fieldIdMain[1].fieldId = 536870913; /* char field */
   fieldIdMain[1].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   strcpy(fieldIdMain[2].queryFromAlias, "perm1");
   fieldIdMain[2].funcId = AR_MULTI_SCHEMA_FUNC_NONE;  /**/
   fieldIdMain[2].fieldId = 536870914; /* integer field */
   getListFields.numItems = 3;
   getListFields.listPtr = fieldIdMain;  /**/
   /* perm1fromItem from item */
   perm1fromItem.type = AR_MULTI_SCHEMA_SCHEMA_NAME;
   strcpy(perm1fromItem.u.schemaName, "InSubqueryPermissions1");
   perm1fromItem.joinType = 0;
   perm1fromItem.joinQual = NULL;
   strcpy(perm1fromItem.queryFromAlias, "perm1");
   /* perm3fromItem from item */
   perm3fromItem.type = AR_MULTI_SCHEMA_SCHEMA_NAME;
   strcpy(perm3fromItem.u.schemaName, "InSubqueryPermissions4");
   perm3fromItem.joinType = 0;
   perm3fromItem.joinQual = NULL;
   strcpy(perm3fromItem.queryFromAlias, "perm3");
   numMatchesPtr = &numMatches;
   /* main query qualifier */
   qualifier.operation = AR_COND_OP_REL_OP;
   qualifier.u.relOp = &relop;
   relop.operation = IN_NOT_IN_op;
   relop.operandLeft.tag = AR_FIELD;
   strcpy(relop.operandLeft.u.fieldId.queryFromAlias, "perm1");
   relop.operandLeft.u.fieldId.fieldId = 536870913;
   relop.operandRight.tag = AR_VALUE_SET_QUERY;
   relop.operandRight.u.valueSetQuery = &valueSetQuery;
   /* set value set query now */
   valueSetQuery.queryFromList.numItems = 1;
   valueSetQuery.queryFromList.listPtr = &perm3fromItem;
   valueSetQuery.fieldId.fieldId = 536870913;
   strcpy(valueSetQuery.fieldId.queryFromAlias, "perm3");
   valueSetQuery.qualifier = NULL;
   result = ARGetListEntryWithMultiSchemaFields (ctrl, &fromList,
         &getListFields, &qualifier, &sortList, AR_START_WITH_FIRST_ENTRY,
         AR_NO_MAX_LIST_RETRIEVE, FALSE,
         NULL, NULL,  /* groupBy, having */
         &msEntryList, numMatchesPtr, status);
}
```

11.4.2 Java example--value set queries

```
// inNotInOption is either RelationalOperationInfo.AR_REL_OP_IN
// or RelationalOperationInfo.AR_REL_OP_NOT_IN
private void queryUsingSubquery(int inNotInOption, ARServerUser arConnection) throws ARException
{
    // Create subquery part first.
    ValueSetQuery subquery = new ValueSetQuery();

    // Add InSubqueryPermissions4 form as the subquery source.
    QuerySourceForm perm4Form = new QuerySourceForm("InSubqueryPermissions4");
    subquery.addFromSource(perm4Form);

    // The subquery's return value.
    subquery.addFromField(536870913, perm4Form);

    // Create regular query.
    RegularQuery query = new RegularQuery();

    // Add InSubqueryPermissions1 as the from source.
    QuerySourceForm perm1Form = new QuerySourceForm("InSubqueryPermissions1");
    query.addFromSource(perm1Form);

    // Set the regular query's return fields.
    query.addFromField(1, perm1Form);
    query.addFromField(536870913, perm1Form);
    query.addFromField(536870914, perm1Form);

    // Set value set query now.
    ArithmeticOrRelationalOperand field =
    new ArithmeticOrRelationalOperand(536870913, perm1Form);
    ArithmeticOrRelationalOperand arethQuery =
    new ArithmeticOrRelationalOperand(subquery);
    RelationalOperationInfo relOp =
    new RelationalOperationInfo(inNotInOption, field, arethQuery);
    QualifierInfo subqueryQual = new QualifierInfo(relOp);
    query.setQualifier(subqueryQual);

    // Call routine.
    List<QuerySourceValues> results =
    arConnection.getListEntryObjects(query, 0, 0, false, null);
}
```

11.4.3 SQL generated by the call--value set queries

This section shows the SQL generated by the preceding value set query.

Note

In this example, an AR System administrator made the API call, and the form did not use row-level security. If a nonadministrator makes the API call and the form uses row-level security, the SQL generated by the call will differ from this example.

```
SELECT J0.C1, J0.C536870913, J0.C536870914
FROM T172 J0
WHERE (J0.C536870913 IN (SELECT J0.C536870913 FROM T171 J0))
ORDER BY J0.C1 ASC
```

11.4.4 Access control fields and value set queries

If the fields that a value set query is instructed to retrieve (**queryFromList** parameter values) have access controls, those access restrictions are added to the value set query's qualifier (**qualifier** parameter).

11.4.5 Using aggregate functions, groupBy clauses, and having clauses

Beginning with BMC Remedy AR System release 7.6.02, you can apply aggregate functions to the **SELECT** statement of a query, along with **GROUPBY** and **HAVING** clauses. These options can help to refine the selection of data retrieved from the database and reduce the need for in-memory processing of large numbers of records.

You can use the **groupBy** parameter in conjunction with an aggregate function in the **getListFields** parameter to define the set of results on which to apply the aggregate function, specifically **MIN**, **MAX**, **SUM**, **AVG**, or **COUNT**.

For example, if the application contains a table of ticket IDs, ticket statuses, and support engineer names, you could use the **COUNT** aggregate to generate an SQL query that would count and return the open tickets the for each support engineer (**agent**):

```
SELECT COUNT(ticketId),agent FROM tickets WHERE status='Open' GROUP BY agent
```

To further refine the results of this query, you could also add a **HAVING** clause. The having parameter can also take an aggregate function. For example, to return records for only those support engineers who have more than ten open tickets, you would add a **HAVING** clause with a **COUNT** function:

```
SELECT COUNT(ticketId),agent FROM tickets WHERE status='Open' GROUP BY agent HAVING
COUNT(ticketId) > 10
```

Note

BMC Remedy AR System releases before 7.6.02 do not support the **groupBy** and **having** options or the aggregate functions. If you send an **ARGetListEntryWithMultiSchemaFields** call with a non-empty **groupBy** or **having** list, or an aggregate function in the **getListFields** parameter, the unrecognized option is stripped from the query. This can produce unexpected query results.

C example and exercise

This example illustrates some sample code that will generate a **SELECT** query using the **COUNT** aggregate function with **GROUP BY** and **HAVING** clauses.

The example code queries the **field_dispprop** form, an **AR System Metadata** form that stores information about display properties for fields and forms, to determine the number of records in which data is stored as a CLOB. For each distinct schema and view, this query determines the total number of display properties and the number of non-NULL short display properties. You could then subtract the short property count from the total to get the number of display properties that are stored as CLOBs rather than as regular text fields in

field_dispprop.

Here is the example code:

```
static void
buildFieldId(ARMultiSchemaFieldIdStruct *ff,
      const char *a, ARInternalId fieldId)
{
   strcpy(ff->queryFromAlias, a);
   ff->fieldId = fieldId;
}
}
static void
buildFieldFunc(ARMultiSchemaFieldFuncStruct *ff,
      const char *a, ARInternalId fieldId, int funcId)
{
   strcpy(ff->queryFromAlias, a);
   ff->fieldId = fieldId;
   ff->funcId = funcId;
}
}
int
arschematest(ARControlStruct *ctrl, ARStatusList *status)
{
   int result = AR_RETURN_OK;
   static const char field_disppropName[]="AR System Metadata: field_dispprop";
   static const char field_disppropAlias[] = "field_dispprop";
   enum {
      F_recordId = 1,
      F_vuiId = 20017,
      F_schemaId = 20018,
      F_propShort = 20504
   };
   ARMultiSchemaFuncQueryFromList fromList = {0,};
   ARMultiSchemaFuncQueryFromStruct fromItem[1];
   unsigned int numMatches = 0, *numMatchesPtr = &numMatches;
   unsigned int numMatches = 0, *numMatchesPtr = &numMatches;
   /*
    * SELECT f.schemaId, f.vuiId, COUNT(f.recordId), COUNT(f.propShort),
    *    FROM field_dispprop f
    *    GROUP BY f.schemaId, f.vuiId
    *    HAVING COUNT(f.propShort) < COUNT(f.recordId)
    */
   fromItem[0].type = AR_MULTI_SCHEMA_SCHEMA_NAME;
   strcpy(fromItem[0].u.schemaName, field_disppropName);
   strcpy(fromItem[0].queryFromAlias, field_disppropAlias);
   fromItem[0].joinType = 0;
   fromItem[0].joinQual = NULL;
   fromList.numItems = 1;
   fromList.listPtr = fromItem;
   fromList.listPtr = fromItem;
   ARMultiSchemaFieldFuncStruct   fieldId[4];
   buildFieldFunc(&fieldId[0], field_disppropAlias,
         F_schemaId, AR_MULTI_SCHEMA_FUNC_NONE);
   buildFieldFunc(&fieldId[1], field_disppropAlias,
         F_vuiId, AR_MULTI_SCHEMA_FUNC_NONE);
   buildFieldFunc(&fieldId[2], field_disppropAlias,
         F_recordId, AR_MULTI_SCHEMA_FUNC_COUNT);
   buildFieldFunc(&fieldId[3], field_disppropAlias,
         F_propShort, AR_MULTI_SCHEMA_FUNC_COUNT);
   ARMultiSchemaFieldFuncList selectList = {0,};
   selectList.numItems = 4;
   selectList.listPtr = fieldId;
   selectList.listPtr = fieldId;
   ARMultiSchemaFieldIdStruct groupBy[2];
   ARMultiSchemaFieldIdList groupByList = {2, groupBy};
   buildFieldId(&groupBy[0], field_disppropAlias, F_schemaId);
   buildFieldId(&groupBy[1], field_disppropAlias, F_vuiId);
   buildFieldId(&groupBy[1], field_disppropAlias, F_vuiId);
   ARMultiSchemaFuncQualifierStruct having = {AR_COND_OP_NONE, NULL};
   ARMultiSchemaFuncRelOpStruct havingRel = {0,};
   having.operation = AR_COND_OP_REL_OP;
   having.u.relOp = &havingRel;
   {
      ARMultiSchemaFieldFuncValueOrArithStruct *ffv = &havingRel.operandLeft;
      havingRel.operation = AR_REL_OP_LESS;
      ffv->tag = AR_FIELD;
      buildFieldFunc(&ffv->u.fieldFunc,
            field_disppropAlias, F_propShort, AR_MULTI_SCHEMA_FUNC_COUNT);
      ffv = &havingRel.operandRight;
      ffv->tag = AR_FIELD;
      buildFieldFunc(&ffv->u.fieldFunc,
            field_disppropAlias, F_recordId, AR_MULTI_SCHEMA_FUNC_COUNT);
   }
   }
   ARMultiSchemaSortList        sortList = {0,};
   ARMultiSchemaFieldFuncValueListList msEntryList = {0, NULL};
   ARMultiSchemaFieldFuncValueListList msEntryList = {0, NULL};
   result = ARGetListEntryWithMultiSchemaFields(ctrl, &fromList,
               &selectList, NULL, &sortList,
               AR_START_WITH_FIRST_ENTRY, AR_NO_MAX_LIST_RETRIEVE,
               FALSE, &groupByList, &having, &msEntryList,
               numMatchesPtr, status);
   return result;
}
```

The generated SQL query is:

```
SELECT J0.C20018, J0.C20017, COUNT(J0.C1), COUNT(J0.C20504) FROM T87 J0 GROUP BY J0.C20018,
J0.C20017 HAVING(COUNT(J0.C20504) < COUNT(J0.C1))
```

As an exercise, join the **AR System Metadata: field_dispprop** form with the **AR System Metadata: arschema** form to retrieve form names instead of or in addition to integer identifiers.

11.5 Vendor form joins

The ARGetListEntryWithMultiSchemaFields function can join multiple vendor forms with any number and type of other forms at runtime.

The function stores data that it retrieves from the vendor data source in a temporary table. It then joins this table with tables for the AR System forms in the **queryFromList** parameter to retrieve the primary SQL query result. Afterward, it clears the temporary table to remove the vendor data from AR System.

For information about vendor forms and their associated ARDBC plug-ins, see Creating vendor forms. The following topics are provided:

- Creating efficient qualifiers for vendor forms
- Configuring the number of temporary tables for vendor forms

11.5.1 Creating efficient qualifiers for vendor forms

Vendor form joins can diminish the performance of the ARGetListEntryWithMultiSchemaFields function. To optimize its performance, the function automatically retrieves only a subset of data from the vendor data source when appropriate.

To avoid retrieving all the data in a vendor data source unless necessary, do not issue unqualified queries to the vendor data source. When constructing qualifiers for a call that includes vendor forms, try to include as many instances of the following conditions as possible:

- *vendorField* = *constantValue*
- *vendorField* = *anotherVendorField*
- *vendorField* = NULL

In some cases, using outer joins in the join qualifier of a vendor form structure can cause the function to ignore the preceding conditions and retrieve all the data in the vendor data source.

11.5.2 Configuring the number of temporary tables for vendor forms

The first time that ARGetListEntryWithMultiSchemaFields queries a vendor form, it creates a temporary table to store the vendor data on the AR System server. By default, only one temporary table for each vendor form can exist on the server.

To process multiple requests from this function to the same vendor form, you can use multiple temporary tables to improve the performance. Use the following procedures to enable multiple temporary tables for each vendor form to exist. You can configure this setting globally and for individual forms. The form setting overrides the global setting.

To configure the number of temporary tables for all vendor forms

This procedure sets the maximum number of temporary tables for *all* vendor forms on the AR System server.

1. Open the AR System Administration: Server Information form.

2. Click the **Advanced** tab.
3. In the **Maximum Vendor Temp Tables** field, enter the maximum number of temporary tables that can exist simultaneously for a single vendor form.
4. Click **Apply** or **OK**.
   The change takes effect immediately. You do not have to restart the server.

To configure the number of temporary tables for a single vendor form in version 8.1 and later of AR System

This procedure sets the maximum number of temporary tables for an individual vendor form. It overrides the global server setting.

1. In BMC Remedy Developer Studio, open the appropriate vendor form.
2. Select the **Definitions** tab in the form editor.
3. Expand the **Other Definitions** panel and then the **Vendor Information** panel.
4. Enter the appropriate integer in the **Maximum Vendor Temp Tables** field.
5. Click **File > Save**.

12 BMC Remedy AR System Java API overview

The BMC Remedy AR System Java API is an application programming interface to integrate with BMC Remedy AR System. This section includes the information you need to start developing your BMC Remedy AR System Java client. The Java API is a collection of classes, interfaces, and relationships that provide full client functionality like the BMC Remedy AR System C API in a style consistent with typical Java programming techniques. Like the C API, the Java API is forward and backward compatible with other versions of BMC Remedy AR System.

Use the following information for details about the BMC Remedy AR System Java API.

- BMC Remedy AR System Java API installed files
- Runtime configuration
- BMC Remedy AR System Java API programming model
- Programming with the Java API
- Java API sample code for managing BMC Remedy AR System records

12.1 BMC Remedy AR System Java API installed files

To install the BMC Remedy AR System Java API, select the API server component when you install the BMC Remedy AR System server. See Installing a server group.

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

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

The BMC Remedy AR System C API files are also in this directory. To verify that the Java API is installed, check that files listed in the Contents of the BMC Remedy BMC Remedy AR System Java API installation section are present.

12.1.1 Contents of the BMC Remedy AR System Java API installation

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

Directory	Component	Description
JavaDriver	**build.xml** **CommandProcessor.java** **Commands.java** **ImageExtractor.java** **InputFile.javaInput** **Reader.java**	Sample Java code files that contain Java API examples.

	JavaDriver.java **LocaleCharSet.java** **OutputWriter.java** **PerfJavaDriver.java** **PerfThreadControlBlock.java** **RandomNumberThread.java** **RowIterator.java** **SyncObject.java** **ThreadControlBlock.java** **ThreadStartInfo.java** **WFD.java** **WfdCommands.java** **WfdOutputWriter.java**	
lib (This directory also contains the required C API library files.)	**apache_axis_license.txt** **apache_crimson_license.txt** **apache_log4j_license.txt** **arapi*VerNum*.dll** 1 **arapi*VerNum*.jar** 1 **arapi*VerNum*.lib** 1 **ardoc*VerNum*.jar** 1 **arjni*VerNum*.dll** 1 **arrpc*VerNum*.dll** 1 **arsys_sample.xml** **arutil*VerNum*.jar** 1 **arutiljni*VerNum*.dll** 1 **arutl*VerNum*.dll** 1 **arxmlutil*VerNum*.dll** 1 **arxmlutil*VerNum*.lib** 1 **axis.jar** **commons-discovery-0.2.jar** **commons-logging-1.0.4.jar** **javadriver.bat** **javadriver.jar** **javawfd.bat** **jaxrpc.jar** **jlicapi*VerNum*.dll** 1 **jlicapi*VerNum*.jar** 1 **log4j-1.2.14.jar** **log4j-1.2.8.jar** **log4j.properties** **log4j.xml** **saaj.jar** **websvc76.jar**	• API files • Supporting JAR and DLL files • XML configuration files • Batch files for the Java driver and the workflow debugger • HTML API documentation generated by Javadoc

	wsdl4j-1.5.1.jar **xercesImpl.jar** **xmlParserAPIs.jar**	

[1] In these file names, the placeholder ***VerNum*** represents the version number of the release.

12.2 Runtime configuration

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

- Java Runtime Environment (JRE) 6 Update 17.
- Required JAR files in the **CLASSPATH** environment variable or the **java** command **-classpath** command-line parameter. Include all the JAR file in the **lib** directory listed in Contents of the BMC Remedy BMC Remedy AR System Java API installation.
- C API library files in the **lib** directory in the correct path:
  - AIX, Linux, and Solaris — **LD_LIBRARY_PATH**
  - HP-UX — **SHLIB_PATH**
  - Windows — **PATH**

Warning

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

12.2.1 Changing the default configuration

To override the default Java API configuration, create the **arsys_api.xml** configuration file and make sure it is in the **CLASSPATH** environment variable or the **java** command **-classpath** command-line parameter. See the sample configuration file, **arsys_sample.xml**, in the **lib** directory for descriptions, valid values, and default values for the API configuration options.

12.2.2 Running the Java driver program

To run the Java driver program, test your environment by running the BMC Remedy AR System server and the JavaDriver program. This program illustrates the capabilities of the BMC Remedy AR System Java API. The JavaDriver program works almost exactly the same as the C driver program.

12.3 BMC Remedy AR System Java API programming model

Consistent with object-oriented design, the BMC Remedy AR System Java API represents BMC Remedy AR System server objects as Java objects. Java classes are defined for forms, fields, menus, active link, filter, escalations, and all other objects in a BMC Remedy AR System application. Entry objects represent entries (requests) so your Java client can manipulate BMC Remedy AR System data as well as definitions. These sections describe the different types of objects and how exceptions are handled:

- ARServerUser object
- Server objects
- Cloning objects
- Exception handling

For more information, see the Java API online documentation HTML files in the **ardoc*VerNum*.jar** file listed in Contents of the BMC Remedy BMC Remedy AR System Java API installation.

To access the Java API documentation, unzip the **ardoc*VerNum*.jar** file to create a tree of directories, then open the **index.html** file to see an overview of the entire BMC Remedy AR System Java API documentation with links to the details. (To unzip a JAR file, use a zip utility the Java **jar** executable, which is in the **bin** directory of the Java JRE is installation. For example, **jar -xvf ardoc*VerNum*.jar**.)

In these file names, the placeholder ***VerNum*** represents the version number of the release.

12.3.1 ARServerUser object

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

12.3.2 Server objects

The BMC Remedy AR System Java API includes classes for all server objects: **Form**, **Field**, **View**, **ActiveLink**, **Escalation**, **Filter**, **Container**, **Menu**, **Entry**, and **SupportFile**. The **Form**, **Field**, **Container**, and **Menu** classes contain subclasses that represent specialized types. For example, **RegularForm**, **IntegerField**, **ApplicationContainer**, and **SqlMenu**.

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

12.3.3 Cloning objects

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

12.3.4 Exception handling

Errors are modeled through the **ARException** class. All error messages that the AR System server returns are generated as an **ARException** in the BMC Remedy AR System Java API. Warnings and informational status messages are not treated as exceptions, but are available using the **getLastStatus** method of the **ARServerUser** class.

12.4 Programming with the Java API

12.4.1 Java API requirements

To build and run a BMC Remedy AR System Java API program on either Windows or UNIX, you need the following environment components:

- J2SE Software Development Kit (SDK), version 5 (1.5.0_06) or higher
- Java API files:
  - **arapi*VerNum*.jar**
  - **arjni*VerNum*.dll/libarjni*VerNum*.a**
  - **arjni*VerNum*.dll/libarjni*VerNum*.sl**
  - **arjni*VerNum*.dll/libarjni*VerNum*.so**
  - **arsys_sample.xml**
  - **log4j.xml**
  - **log4j-1.2.14.jar**
  - **websvcjava*VerNum*.jar**

In these file names, the placeholder ***VerNum*** represents the version number of the release. In some cases, this includes a build number. For example, in release 7.6.02, the AR System API file is named **arapi7602.jar** or **arapi7602_build001.jar**.

In addition to the above mentioned **arapi*VerNum*.jar** file, the BMC Remedy AR System Java API file is also available as an OSGI bundle, with the file name as **com.bmc.arsys.api*VerNum*.jar**. The BMC Remedy AR System API OSGI bundle can be utilized in an OSGI deployment environment.

Note

The BMC Remedy AR System Java API also uses a Java Native Interface (JNI) library and the C API library to connect to a pre-7.0.01 BMC Remedy AR System server. You can control when the JNI library is used by setting the **jniLoadMode** parameter in the Java API configuration file. See the comments in the sample file, **arsys_sample.xml**.

12.4.2 To write the Java program

Follow these steps to write the program:

1. Ensure your programming environment is set up correctly as described in the Java API requirements section.
2. Create a Java project in your IDE.
3. Include the **arapi*VerNum*.jar** and the other required JAR files in the BMC Remedy AR System API **lib** directory in the class path.
4. Create a new class for the methods that call the Java API.
5. Import the **com.bmc.arsys.api** package and other packages you might use in your program. The API uses collection classes, so you are likely to need **java.util.ArrayList**, **java.util.List**, and **java.util.Map** .
6. Instantiate an **ARServerUser** object. Set the user name, password, server, and other connection attributes. If the program needs to interact with different servers or as different users, it can create more than one **ARServerUser** object.
7. Use the **ARServerUserlogin** method to open the connection to the server.
8. Perform the required operations using the **ARServerUser** methods and other API objects and methods to create, retrieve, update, and delete BMC Remedy AR System system objects as needed and creating criteria objects and using server object methods as required.
9. Use the **ARServerUserlogout** method to close the connection to the server.

For BMC Remedy AR System Java API troubleshooting information, see Troubleshooting the Java API.

12.4.3 Troubleshooting the Java API

Use these techniques to find errors in your BMC Remedy AR System Java API program:

- **Program fails to start** — Ensure all Java API and application-dependent JAR files are in the classpath.
- **Logging** — Configure logging using the **log4j.xml** file.

12.5 Java API sample code for managing BMC Remedy AR System records

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

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

import com.bmc.arsys.api.*;

import java.util.*;

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

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

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

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

        // Create an entry in a form using the given field values.
        public String createEntry (String submitter, String status,
            String shortDesc) {
            String entryIdOut= "";
            try {
```

```
            Entry entry = new Entry();
            entry.put(Constants.AR_CORE_SUBMITTER, new Value(submitter));
        entry.put(Constants.AR_CORE_STATUS,
            new Value(status, DataType.ENUM));
        entry.put(Constants.AR_CORE_SHORT_DESCRIPTION,
            new Value(shortDesc));
        entryIdOut = server.createEntry(formName, entry);
        System.out.println();
        System.out.println("Entry created. The id # is " +
            entryIdOut);
    } catch (ARException e) {
        ARExceptionHandler(e, "Cannot create the entry." );
    }
    return entryIdOut;
}

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

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

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

System.out.print(callDateTimeTS.toDate());

                                                }
                                            System.out.println("");
                }
            } catch( ARException e ){
                ARExceptionHandler (e,
                                        "Problem while querying by entry ID.");
            }
        }

        // Retrieve entries from the form using the given qualification. With
        // the returned entry set, print out the ID of each entry and the
        // contents in its shortDescription field.
        void queryEntrysByQual(String qualStr) {
            System.out.println();
            System.out.println ("Retrieving entryies with qualification " +
                qualStr);
            try {
                     // Retrieve the detail info of all fields from the form.
                List <Field> fields =
server.getListFieldObjects(formName);
                // Create the search qualifier.
                QualifierInfo qual = server.parseQualification(qualStr,
                                        fields, null, Constants.AR_QUALCONTEXT_DEFAULT);

                int[] fieldIds = {2, 7, 8};
                OutputInteger nMatches = new OutputInteger();
                List<SortInfo> sortOrder = new ArrayList<SortInfo>();
                sortOrder.add(new SortInfo(2,
Constants.AR_SORT_DESCENDING));
                // Retrieve entries from the form using the given
                // qualification.
                List<Entry> entryList = server.getListEntryObjects(
                                               formName, qual, 0,
Constants.AR_NO_MAX_LIST_RETRIEVE,
                               sortOrder, fieldIds, true, nMatches);

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

entryList.get(i).get(Constants.AR_CORE_SHORT_DESCRIPTION));
                                            }
```

```
                  }
             } catch( ARException e ) {
                    ARExceptionHandler(e,
                                 "Problem while querying by qualifier. ");
             }
         }

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

         public void printStatusList(List<StatusInfo> statusList) {
             if (statusList == null || statusList.size()==0) {
                  System.out.println("Status List is empty.");
                      return;
             }
             System.out.print("Message type: ");
             switch(statusList.get(0).getMessageType())
             {
                 case Constants.AR_RETURN_OK:
                                   System.out.println("Note");
                             break;
                 case Constants.AR_RETURN_WARNING:
                              System.out.println("Warning");
                              break;
                 case Constants.AR_RETURN_ERROR:
                              System.out.println("Error");
                                  break;
                 case Constants.AR_RETURN_FATAL:
                                       System.out.println("Fatal Error");
                            break;
                 default:
                            System.out.println("Unknown (" +
                                        statusList.get(0).getMessageType() + ")");
                              break;
             }
             System.out.println("Status List:");
             for (int i=0; i < statusList.size(); i++) {

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

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

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