

HP OpenView Service Desk 3.0

API Programmer's Guide

Second Edition



Manufacturing Part Number: N/A

June 2000

Legal Notices

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19 (c)(1,2).

Copyright Notice. © Copyright 2000 Hewlett-Packard Company

Trademark Notices

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

Contents

1. The API

API Overview	14
Runtime Architecture	14
Requirements	16
Running API-based programs	16
Developing API-based programs	17

2. API Principles

Principle API Functions	22
Connecting to the Application Server	22
Identifying Entities and Attributes	22
Finding the Proper Entity Instantiation	23
Getting an Entity Instantiation	23
Getting Attribute Values	24
Setting Attribute Values	24
Saving Instantiations	24
Single Instantiation Processing	25
Error Presentation	25

3. Examples

Installing the Examples	28
Connecting to the Application Server	29
Identifying Entities and Attributes	31
Finding the Proper Entity Instantiation	36
Getting an Entity Instantiation	42
Creating New Objects	45
Getting and Setting Attribute Values	47
Saving Instantiations	51

Contents

Single Instantiation Processing.....	54
Error Presentation.....	58
4. Javadoc	
Javadoc Elements.....	62
Glossary	

Preface

The Service Desk **API** is an application programming interface designed to give developers programmatic access to the Service Desk application. By using the API, you can create customized integrations with Service Desk.

This guide contains information about the structure and potential use of the Service Desk API. It is a technical guide designed for use by experienced Java application programmers.

NOTE

You must have knowledge and experience programming with MS Visual J++ to develop programs using the Service Desk API and this guide. The Service Desk API is formed by a combination of Java classes written in MS Visual J++ 6.0.

This guide is organized as follows:

- Chapter 1, “The API,” on page 13 provides an overview of the API, the Service Desk architecture, and a list of requirements for using the API.
- Chapter 2, “API Principles,” on page 19 briefly describes basic functions available with the API.
- Chapter 3, “Examples,” on page 27 provides explanations and examples of the most common API functions.
- Chapter 4, “Javadoc” on page 61 contains the Javadoc reference, which is a computer-generated API output file.
- The “Glossary” on page 137 provides a list of terms which may be unfamiliar to you, with definitions.

Related Publications

This section may help you find information that is related to the information in this guide. It gives an overview of the Service Desk documentation and lists other publications you may need to refer to when using this guide.

The Service Desk Documentation

Service Desk provides a selection of books and online help to assist you in using Service Desk and improve your understanding of the underlying concepts. This section illustrates what information is available and where you can find it.

- The `Readme.htm` file on the Service Desk CD-ROM contains information that will help you get started with Service Desk. It also contains any last-minute information that became available after the other documentation went to manufacturing.
- The *HP OpenView Service Desk: Release Notes* give a description of the features that Service Desk provides. In addition, they give information that helps you:
 - compare the current software's features with those available in previous versions of the software;
 - solve known problems.

The Release Notes are available as a PDF file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Release_Notes.pdf`.

- The *HP OpenView Service Desk: Supported Platforms List* contains information that helps you determine platform and software requirements and compatibility. It lists the combinations of platforms and software Service Desk 3.0 was tested on.

The Supported Platforms List is available as an HTML file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Supported_Platforms_List.htm`.

- The *HP OpenView Service Desk: Installation Guide* covers all aspects of installing Service Desk.

The Installation Guide is available as a PDF file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Installation_Guide.pdf`.

- The *HP OpenView Service Desk: Data Exchange Administrator's Guide* explains how you can use data from other applications in Service Desk. It explains the underlying concepts of the data exchange process and gives step-by-step instructions on exporting data from external applications and importing it into Service Desk. The data exchange process includes importing single service events and batches of data.

The Data Exchange Administrator's Guide is available as a PDF file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Data_Exchange.pdf`.

- The *HP OpenView Service Desk: API Programmer's Guide* contains information that will help you create customized integrations with Service Desk. This guide depicts the API structure, and explains some of the basic functions with examples for using the Application Programming Interface (API) provided with Service Desk. The API extends the HP OpenView Service Desk environment by providing independent programmatic access to data-centered functionality in the Service Desk application server environment.

The API Guide is available as a PDF file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `API_pg.pdf`.

- The *HP OpenView Service Desk: Data Dictionary* contains helpful information about the structure of the application.

The Data Dictionary is available as an HTML file on the HP OpenView Service Desk 3.0 CD-ROM. The file name is `Data_Dictionary.htm`.

- The online help is an extensive information system providing:
 - procedural information to help you perform tasks, whether you are a novice or an experienced user;
 - background and overview information to help you improve your understanding of the underlying concepts and structure of Service Desk;
 - information about error messages that may appear when working with Service Desk, together with information on solving these errors;
 - help on help to learn more about the online help.

The online help is automatically installed as part of the Service Desk application and can be invoked from within Service Desk. See the

following section entitled “Using the Online Help” for more information.



Reading PDF Files

You can view and print the PDF files with Adobe® Acrobat® Reader. This software is included on the HP OpenView Service Desk 3.0 CD-ROM. For installation instructions, see the `readme.htm` file on the CD-ROM.

The latest version of Adobe Acrobat Reader is also freely available from Adobe’s Internet site at <http://www.adobe.com>.

Using the Online Help

You can invoke help from within Service Desk in the following ways:

- To get help for the window or dialog box you are working in, do one of the following:
 - Press **F1**.
 - Click the help toolbar button .
 - Choose **Help** from the **Help** menu.
 - Click the help command button  in a dialog box.
- To search for help on a specific subject using the table of contents or the index of the help system: choose **Help Contents & Index** from the **Help** menu.

When you are in the help viewer, you can find help on how to use the help system itself by clicking the **Help** toolbar button:






Service Desk also provides *tooltips* and “*What’s This?*” help for screen items like buttons, boxes, and menus.

A *tooltip* is a short description of a screen item. To view a tooltip, rest the mouse pointer on the screen item. The tooltip will appear at the position of the mouse pointer.

“*What’s This?*” help is a brief explanation of how to use a screen item. “*What’s this?*” help generally gives more information than tooltips. To view “*What’s This?*” help:

1. First activate the “*What’s This?*” mouse pointer in one of the following ways:

- Press **Shift+F1**.
- Click the “What’s this?” toolbar button .
- Choose What’s This? from the Help menu.
- In dialog boxes, click the question mark button  in the title bar.

The mouse pointer changes to a “What This?” mouse pointer .

2. Then click the screen item for which you want information. The “What’s This?” help information appears in a pop-up window.

To close the “What’s This?” pop-up window, click anywhere on the screen or press any key on your keyboard.

Other Related Publications

In addition to the Service Desk documentation mentioned above, you may want to refer to the following publications when using this guide:

- <http://java.sun.com/products/jdk/javadoc/index.html>. Link connects to the Java™ home Web site.
- <http://msdn.microsoft.com/visualj/default.asp>. The home Web site for Microsoft® Visual J++.
- <http://www.microsoft.com/java/sdk/>. This is the site for Microsoft SDK for Java.

Typographic Conventions

The table below illustrates the typographic conventions used in this guide.

Font	What the Font Represents	Example
<i>Italic</i>	References to book titles Emphasized text	See also the <i>HP OpenView Service Desk: Installation Guide</i> . <i>Do not delete</i> the System user.
Bold	First-time use of a term that is explained in the glossary	The service call is the basis for incident registration.
<code>Courier</code>	Menu names Menu commands Button names File names Computer-generated output, such as command lines and program listings	You can adjust the data view with the commands in the View menu. Choose Save from the menu. Click Add to open the Add Service Call dialog box. To start the installation, double-click setup.htm. If the system displays the text C:\>dir a: The device is not ready then check if the disk is placed in the disk drive.
Courier bold	User input: text that you must enter in a box or after a command line	If the service call must be solved within 30 minutes, enter 30 .
<i>Courier italic</i>	Replaceable text: text that you must replace by the text that is appropriate for your situation	Go to the folder X:\Setup, where X is your CD-ROM drive.
Helvetica bold	Keyboard keys A plus sign (+) means you must press the first key (Ctrl in the example), hold it, and then press the second key (F1 in the example).	Press Ctrl+F1 .

We Welcome Your Comments!

Your comments and suggestions help us understand your needs, and better meet them. We are interested in what you think of this manual and invite you to alert us to problems or suggest improvements. You can submit your comments through the Internet, using the HP OpenView Documentation Comments Web site at the following URL:

http://ovweb.external.hp.com/lpe/comm_serv

If you encounter *serious errors* that impair your ability to use the product, please contact the HP Response Center or your support representative.

The latest versions of OpenView product manuals, including Service Desk manuals, are available on the HP OpenView Manuals Web site at the following URL:

http://ovweb.external.hp.com/lpe/doc_serv

Software patches and documentation updates that occur after a product release, will be available on the HP OpenView Patches Web site at the following URL:

<http://ovweb.external.hp.com/cpe/patches>

1 **The API**

This chapter provides an overview of how the API fits into the Service Desk architecture. It also lists the requirements which must be met to use the Service Desk API effectively.

API Overview

The Service Desk API is a set of Java classes, written in Visual J++. These classes are part of the standard Service Desk software bundle. The API provides a means of integrating with the Service Desk application server independently from the user interface, while ensuring that all authorization and **business rules** are enforced. You can use this API to integrate other management tools with Service Desk. For example, an inventory tool could be integrated so that it automatically updates the Service Desk database when new items or changes in existing items are found by the external inventory tool. The Data Exchange feature included with the Service Desk application was created using this API, for example.

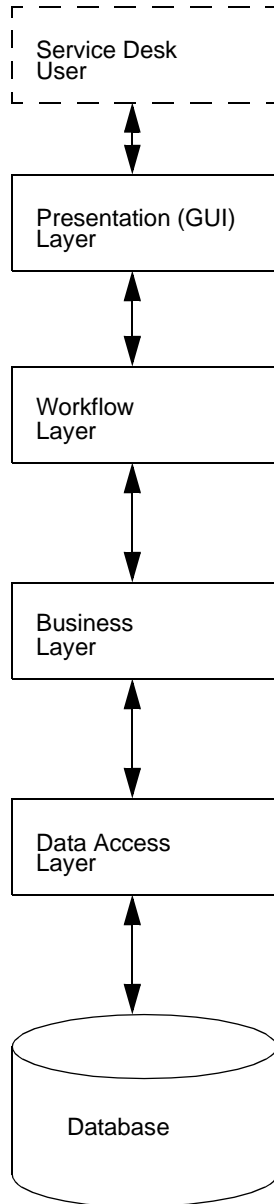
For information on the basic functions performed with the API, see Chapter 2, “API Principles,” on page 19, or Chapter 3, “Examples,” on page 27. To view the Javadoc document containing: classes, interfaces, constructors, methods and fields see Chapter 4, “Javadoc,” on page 61.

Runtime Architecture

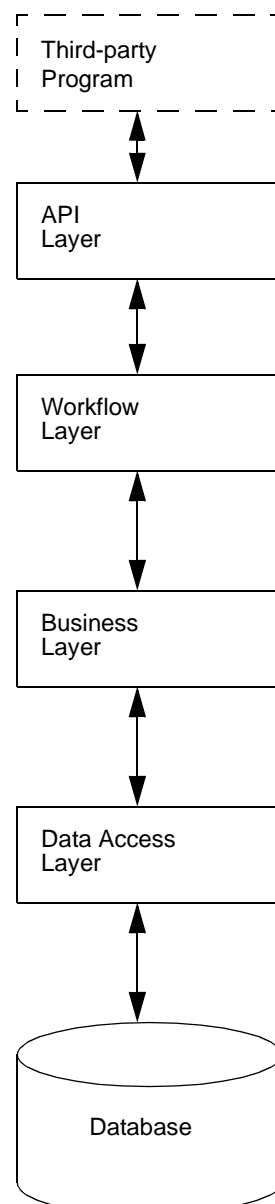
The API is tightly integrated with the Service Desk architecture. Figure 1-1 on page 15 shows how the API fits into this architecture. The Service Desk application architecture is made up of four layers, with each layer performing a specific task:

- The **presentation layer** contains the user interface. This layer is in effect replaced by the API.
- The **workflow layer** contains the rules that determine what information is required to complete an operation.
- The **business layer** contains rules that determine how an operation is validated or completed. It communicates with the data access layer.
- The **data access layer** provides access to the data in the database. It communicates between the application’s **object model** and the relational representation of what exists in the underlying database.

Figure 1-1
Service Desk Architecture
Without API Integrated



With API Integrated



End users normally communicate with the Service Desk application

The API

API Overview

through the graphical user interface (GUI) referred to as the presentation layer shown on the left in Figure 1-1 on page 15. The API takes the place of the presentation layer, while the role of the user is replaced by a third-party program developed to interact with the API. This is depicted on the right side of Figure 1-1 on page 15. The third-party application provides input to Service Desk while following a number of self-defined rules to ensure the input will be processed appropriately. The API is capable of communicating with the application server over the workflow layer with the same result as if the actions were initiated from the user interface. The difference is that the API reacts to input from another application, causing Service Desk to perform an action.

The API depends heavily on the availability of the Service Desk workflow layer for access to the Service Desk environment. The workflow layer assures that all rules normally applied to actions in the user interface are also applied to the same actions when they are performed by the API. For example, rules exist in the workflow layer to ensure that a service call can never have an end date before the start date. The business and workflow layers work together to make a business object fully functional.

Requirements

The Service Desk API can be run from both the server or client environment. The **classes** that form the API are stored in the `classes.zip` file installed with the Service Desk application. The installation procedure adjusts the class path to include the API package by default. Additional requirements for successful use of the API include:

Running API-based programs

Requirements for running API-based programs:

- The `classes.zip` file must be referred to in the local class path setting. If you have installed Service Pack 1, a reference must be made to the `sd_sp1.zip` file.
- Programs must be run from a machine that is set up as a Service Desk application server, for more information refer to the *HP OpenView Service Desk Installation Guide*.
- A Service Desk account needs to be created providing access to all applicable areas. It can be a **non-UI account**, see page 139 for more information.

Developing API-based programs

Requirements for developing API-based programs:

- Programming knowledge of MS Visual J++.
- MS Visual J++ 6.0, or an equivalent development environment. For example, you can compile with the Microsoft Software Development Kit (SDK) and then run it with a Microsoft Jview virtual machine. Jview is installed with Service Desk by default. More information about the Microsoft SDK is available at the following Web site:
<http://www.microsoft.com/java/sdk/>
- Reference to the Service Desk `classes.zip` delivered with the Service Desk application in the IDE's class path setting. If you have installed Service Pack 1 you should reference the `sd_sp1.zip` file. It can be downloaded from the following Web site:
<http://ovweb.external.hp.com/cpe/patches/>

For information on downloading the examples, the javadoc, and this guide, see "Installing the Examples" on page 28.

The Service Desk API is formed by a number of Java classes written in Microsoft Visual J++ 6.0. These classes indirectly use some Microsoft-specific extensions of the Java language. Their inter-operability with non-Microsoft virtual machines and compilers is not guaranteed.

The API
API Overview

2 **API Principles**

The API provides users with access to data-related functions in the Service Desk environment. It forms a layer in the software environment with the purpose of assuring optimum communication between the entities on either side:

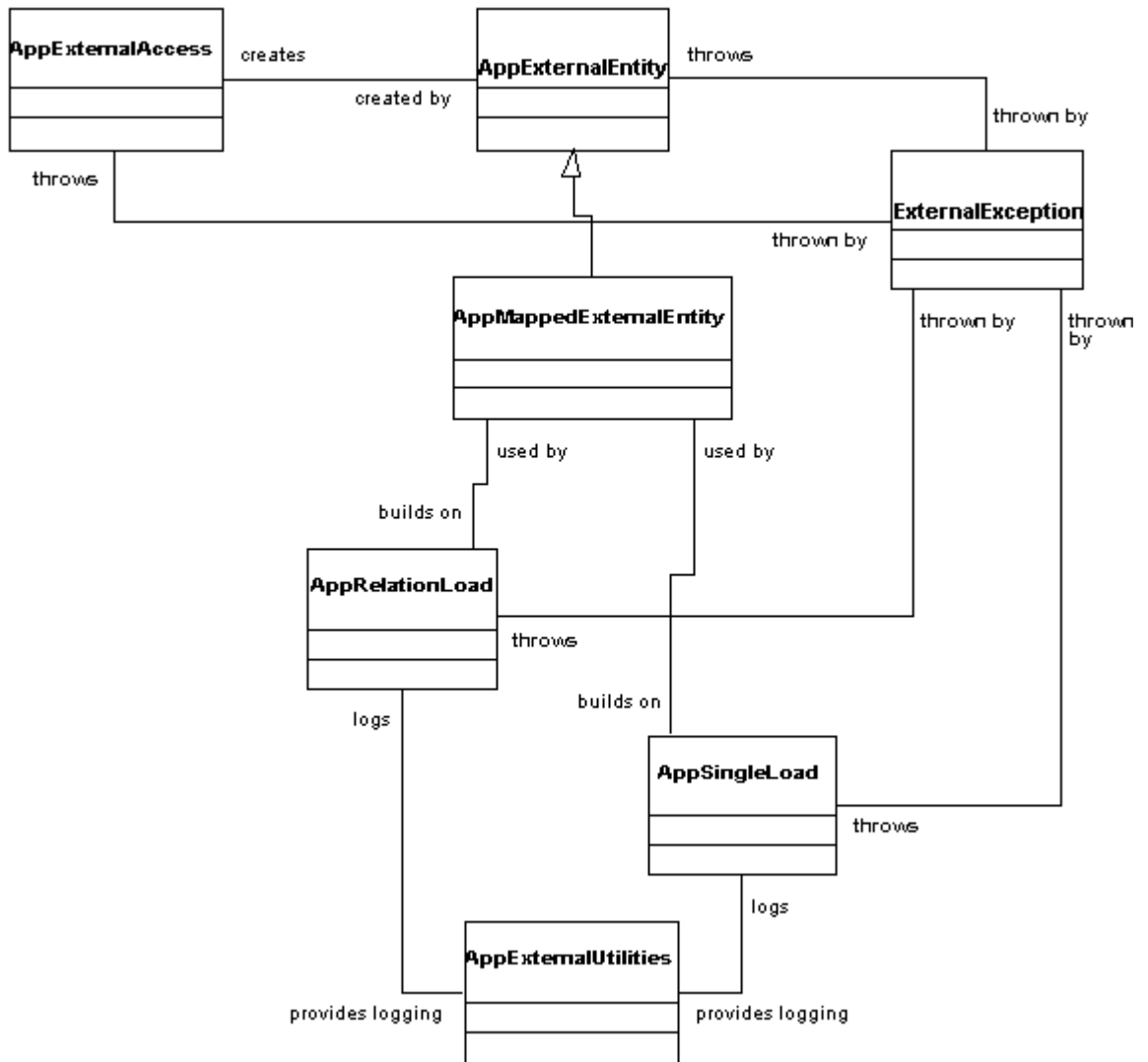
- The third-party program performing actions on Service Desk entities
- The Service Desk application server giving access to Service Desk entities

The Service Desk application is created around a **kernel**, which is defined as a collection of generic reusable sources called ITSM Foundation Classes (IFC). The generic elements that make up the kernel come from all layers of the application. The API is included in the kernel. Because Service Desk-specific functions do not exist in the **IFC**, the IFC can be used for developing other database-dependent applications. The primary classes that form the API are in the `com.hp.ifc.ext` package. You will also need classes from `com.hp.ifc.util.marshall` to communicate with the workflow layer, and specific object classes from `com.hp.ifc.types`.

An additional supporting package within the kernel is `com.hp.ifc.rep.ext`. This package contains the supporting classes that define mapping to external data sources used by the API classes. This package is part of the **repository**, see page 139.

The following object model shows how API classes work together. It is followed by sections explaining the most commonly used API functions:

Figure 2-1 Main API Functions



Principle API Functions

The following sections briefly explain some of the primary API functions. A more detailed explanation together with examples can be found in Chapter 3, “Examples,” on page 27.

Connecting to the Application Server

To obtain access to functions available from the workflow layer, all programmatic API sessions must first log on. API sessions log on using a specialized class, `AppExternalAccess` that also operates as an object server for Service Desk, creating and retrieving objects from the Service Desk environment as needed. For example, retrieving a service call so that it can be modified.

The class `AppExternalAccess` provides access to data managed by the Service Desk application server. This class can be considered a gateway into the workflow layer for all requests going to the application server. Every program that uses the API will be organized around at least one `AppExternalAccess` object to establish a connection with the application server. After successfully logging on, this same object functions as a Service Desk entity server. It takes care of:

- logging on and off from the Service Desk environment;
- activating import mapping settings and presenting those settings;
- creating new Service Desk objects and retrieving existing objects for further processing;
- presenting warnings and error messages created by the Service Desk environment.

For more information, including a detailed example, see “Connecting to the Application Server” on page 29.

Identifying Entities and Attributes

An important part of the integrating with Service Desk involves the manipulation of Service Desk entities and their attributes. A reliable method for identifying entities and attributes must be available for that purpose. The API uses two means of identifying Service Desk entities and attributes for processing:

- By communicating with the application server, creating or modifying Service Desk entities and attributes as necessary. The enumeration class `ITSMExternalEnumeration` identifies the objects and attributes by means of the numeric values assigned to them in the Service Desk repository. This means of identification is also referred to as reference by number. Reference by number provides a high degree of freedom when manipulating Service Desk entities.
- By communicating with the application server using pre-defined import mapping settings. Import mapping requires setup by the Service Desk application administrator prior to actually programming the integration. These settings are labels defined by the user for entities and attributes. Once defined, the labels can be used within the Service Desk environment. Using labels to access data is referred to as reference by label. In order to access the settings, the active `AppExternalAccess` object must be set to use the proper group of import mapping settings.

For more information, including a detailed example, see “Identifying Entities and Attributes” on page 31.

Finding the Proper Entity Instantiation

Once a relevant Service Desk entity is identified, the proper instantiations of that entity must be located. The `AppExternalAccess` class contains methods for the construction of search criteria, using either the reference by number or the reference by label methods. These criteria can then be used in an `AppExternalAccess` method that returns a list of identifiers for the objects that comply with the search conditions.

For more information, including a detailed example, see “Finding the Proper Entity Instantiation” on page 36.

Getting an Entity Instantiation

After obtaining the **entity** instantiation identifier, the actual data can be retrieved from the Service Desk application server. `AppExternalAccess` offers a number of methods for retrieving the data. Methods differ mainly in the way they reference entities and attributes. When reference by number is used, the method returns an `AppExternalEntity` object. When reference by label is used, an `AppMappedExternalEntity` object is returned. `AppMappedExternalEntity` is an extension of the `AppExternalEntity` class.

Principle API Functions

The `AppExternalAccess` also offers methods for the creation of new Service Desk objects. When used with the `AppMappedExternalEntity` class, the template defined in the import mapping settings and its default values are applied immediately.

For more information, including a detailed example, see “Getting an Entity Instantiation” on page 42.

Getting Attribute Values

After obtaining an `AppExternalEntity` object wrapping Service Desk data, the current attribute values can be determined. When you use the methods defined in `AppExternalEntity` class, these values are returned as Java objects. The objects are either standard Java classes, `java.lang.String` for example, or IFC classes from the `com.hp.ifc.types` package. When using the methods in `AppMappedExternalEntity`, a `String` object is returned for all values.

For more information, including a detailed example, see “Getting and Setting Attribute Values” on page 47.

Setting Attribute Values

The attribute values for an `AppExternalEntity` object can be also be set. You can either send Java objects directly to the API with `AppExternalEntity` using the reference by number method or send strings with `AppMappedExternalEntity` using the reference by label method. Strings are converted into objects and passed to the Service Desk application by the API. With both reference methods, all business rules defined for the object will be applied by the workflow layer.

For more information, including a detailed example see “Getting and Setting Attribute Values” on page 47.

Saving Instantiations

Once the correct data is wrapped in an `AppExternalEntity` object, it can be saved to the database. This will create a new object in the Service Desk environment, or change an existing one, as existing or new objects are processed. While saving the data, all current business rules applicable for that Service Desk entity will be applied.

For more information, including a detailed example, see “Saving Instantiations” on page 51.

Single Instantiation Processing

The separate actions outlined in the sections above offer maximum control over the functionality realized using the API classes. For integrations limited to relatively simple operations, utility classes are provided. The utility actions are a combination of the actions described in the preceding sections. These are `AppSingleLoad`, for Service Desk entities, and `AppRelationLoad`, for relations between Service Desk entities. Both classes use the reference-by-label method. These utilities are capable of:

- adding an object;
- removing an object;
- setting object values;
- adding relations;
- removing relations.

For more information, including a detailed example, see “Single Instantiation Processing” on page 54.

Error Presentation

Exceptions raised within the Service Desk environment usually result in a Microsoft-specific `ComFailException`. These exceptions can be converted to `ExternalException` by the `AppExternalAccess` class. Exceptions are easier to manipulate in the standardized environment provided by the `ExternalException` class.

For more information, including a detailed example, see “Error Presentation” on page 58.

API Principles
Principle API Functions

3 **Examples**

Most example classes explain two methods of referencing Service Desk entities and attributes. Most methods in the example files are defined as static. This has no relevance on the examples, but is intended for easy use within a test class.

Installing the Examples

All example classes provided use the API classes and rely on the contents of the Service Desk demo database. The demo database is installed when you install the evaluation version of Service Desk. When working with the full client version of Service Desk, you will need to select the demo database option during installation.

In order to use the examples they must be compiled and run from the Service Desk *API documentation and examples* patch delivered as `SDSK-00003.exe`. The patch is located at <http://ovweb.external.hp.com/cpe/patches/>. This self-extracting zip file contains:

- the *HP OpenView Service Desk: API Programmer's Guide*;
- the code for the examples explained in this chapter;
- the javadoc for the API classes in HTML format;
- an executable that adjusts the database for use with the examples.

After unzipping the `SDSK-00003.exe` patch, and copying the examples, the following procedure must be completed for the examples to work:

- Step 1.** Compile the example classes, using Microsoft Visual J++ compiler.
- Step 2.** Add the location of the compiled example classes to the current class path. The examples are located in the `com.hp.ifc.ext.example` package. For example, if the compiled classes are located in `C:\Program Files\Service Desk\classes\com\hp\ifc\ext\examples`, the class path should list `C:\Program Files \Service Desk\classes` as one of its entries.
- Step 3.** Run the `CreateStatus` executable. This executable adds some status values to the demo database that are essential when using the examples. A valid Service Desk user name, password, and server will be needed. The syntax is: `CreateStatus <username> <password> <server>`. This program requires the same classpath set for normal Service Desk operations.

Connecting to the Application Server

The `AppExternalAccess` class provides access to data managed by the Service Desk application server. This class can be considered a gateway into the workflow layer for all requests going to the application server. Every program that uses the API will be organized around at least one `AppExternalAccess` object to establish a connection with the application server. After successfully logging on, this same object functions as a Service Desk entity server. The `Example1` class shows how to make a connection and how to verify that a valid connection exists.

The easiest way to log on is with the `AppExternalAccess(String, String, String)` constructor. This uses a user name, password, and server as arguments. These credentials can represent any Service Desk account, including a **non-UI account**. The use of this method is shown in the `Example1.connect()` method. The actual connection is made with the code in line 17:

```
access = new AppExternalAccess(username, password, server);
```

The rest of this method ensures that a message is displayed when a connection is not made.

The `connected()` method shows how to check whether an `AppExternalAccess` object contains a valid connection or not. The essential work is being done by calling `AppExternalAccess.loggedIn()` at line 25. This method returns a boolean value, indicating whether a valid connection exists or not. In the `connected()` method the boolean value determines what message to print.

Example 3-1

Connecting to the Application Server

```
package com.hp.ifc.ext.examples;

import com.hp.ifc.ext.*;

/**
 * Example:
 *   connecting to the HP OpenView Service Desk environment
 *   and testing the connection
 */

public class Example1 {
```

Examples

Connecting to the Application Server

```
private AppExternalAccess access;

public void connect
( String username
, String password
, String server
) {
    try {
        access =
            new AppExternalAccess(username, password, server);
    }
    catch (ExternalException eEx) {
        System.out.println(eEx.getMessage());
    }
}

public void connected() {
    if ((access != null) && (access.loggedIn()))
        System.out.print("Connected to ");
    else
        System.out.print("Not connected to ");
    System.out.println
        ("the Service Desk application server.");
}

public void disconnect() {
    if (access != null) access.disconnect();
}

public void finish() {
    if (access != null) access.shutdown();
}
}
```

Identifying Entities and Attributes

An important part of the API programmer's work involves the manipulation of Service Desk entities and their attributes. A reliable method for identifying entities and attributes must be available for that purpose. The `Example2` class shows two ways of doing this: one using reference by number, the other reference by label. This class also provides an example for extracting the labels that are used when referencing by label.

The most reliable way to reference a Service Desk object is the use of the `ITSMExternalEnum` class. This class is a collection of nested classes, each representing a Service Desk entity. These inner classes are named after the entity represented. For example, the class for the person entity is called `PersonDefEnum`. Each inner class consists of a number of data members, one for each attribute. These data members are named after the attributes, with 'at' as a prefix. 'atBirthDate' thus denotes the `BirthDate` attribute. An `enOid` member is also defined for every inner class. The `enOid` member denotes the numeric value (oid) used to identify the entity. Two simple examples:

```
ITSMExternalEnum.IncidentDefEnum.enOid denotes the incident  
entity. ITSMExternalEnum.IncidentDefEnum.atDescription denotes  
the description attribute for the incident entity.
```

Not every reference can be made easily with a long value, more complex ways also exist. These occur with the aggregation and referencing connections.

Aggregation is a means of referencing when an object is incorporated in another object. In this situation the attributes of the aggregated object can be considered nested objects. An example would be assignment data for an incident. Assignment, when considered as an object, has attributes such as a reference number or an assignment status. These attributes are referenced using long values, that can be specified using the `ITSMExternalEnum` class. For example:

```
long[] ref = new long[]  
{ITSMExternalEnum.IncidentDefEnum.atAssignment  
,ITSMExternalEnum.AssignmentDefEnum.atReferenceNumber};  
denotes the reference number attribute for an incident's assignment.
```

In other cases an object is not aggregated, but referenced. The referenced

Identifying Entities and Attributes

and referencing objects have an independent existence, but they are also linked to each other. For example, when a configuration item is mentioned in connection with an incident.

The `getEntityAndAttributeLongs()` shows how references for an entity and two of its attributes are defined. It returns long values that are referred to by the attributes used. When only indicating a reference, it is enough to give the attribute that references the other object. For example:

`ITSMEExternalEnum.IncidentDefEnum.atConfigurationItem` denotes the reference to a configuration item attribute from the incident entity.

When working with Data Exchange, labels can be defined for Service Desk entities and attributes. These are grouped into import mapping settings (For detailed information on import mapping settings refer to the *HP OpenView Data Exchange Administrator's Guide*). The advantages of using these labels are:

- A tight integration with the labels used for Data Exchange is possible.
- Default values for new objects are defined. These defaults, organized in Service Desk templates, are applied automatically when using reference by label.
- Provides more freedom when choosing label names.

The `getEntityAndAttributeNames()` shows how references for an entity and two of its attributes are defined. As in `Example1`, a connection must be made to the Service Desk application server first. Next, the import settings must be chosen, as is done on line 25:

```
access.setSettings("external_event");
```

The `getEntityAndAttributeNames()` method returns an array of Strings shown in the code. It illustrates how reference by label relies on an external source (like a programmer's memory) to provide the right labels. The methods that use those labels will generate an exception when an invalid label is passed.

The `showAvailableSettings()` method demonstrates means of retrieving the labels defined for import settings, entities and attributes in the Service Desk environment. The values retrieved are presented in an ordered form, but this is purely meant as an example. The main reason to include this method is to demonstrate the methods that can retrieve these labels.

The important lines of code are:

- `String[] settings=access.getAvailableLoadSettingNames();` (line 41). This method call returns an array of all labels defined for import mapping.
- `String[] items = access.getMappedEntityNames();` (line 49). This line results in array of all labels defined for Service Desk entities within a given set of import mapping settings. For the method call to be effective, the `AppExternalAccess` object must first be set to the right import mapping setting with:
`access.setSettings(settings[i]);` (line 47)
- `String[] attrib=access.getMappedAttributeName(items[j]);` (line 55) Finally, this line produces an array of all labels defined for attributes of a Service Desk entity within an import mapping setting.

To print the available settings in alphabetical order the retrieval labels are organized in nested hashtables. This data structure, or a comparable one, can be used to get an overview of the available labels. Also, this makes it possible to check the validity of a label early on.

Apart from the actual labels, you can also retrieve object IDs. These can be obtained as arrays of long values or as arrays of `AppOID` objects, as described above.

Example 3-2 Identifying Entities and Attributes

```
package com.hp.ifc.ext.examples;

import java.util.*;
import com.hp.ifc.ext.*;
import com.hp.ifc.rep.ext.*;

/**
 * Example: referencing entities and attributes
 */

public class Example2 {

    public static long[] getEntityAndAttributeLongs() {
        return new long[]
        { ITSMExternalEnum.IncidentDefEnum.enOid
          , ITSMExternalEnum.IncidentDefEnum.atStatus
          , ITSMExternalEnum.StatusIncidentDefEnum.atText
        };
    }
}
```

Examples

Identifying Entities and Attributes

```
public static String[] getEntityAndAttributeNamees
( String username
, String password
, String server
) {
AppExternalAccess access =
    new AppExternalAccess(username, password, server);
access.setSettings("external_event");
AppExternalEntityInfo ai =
    access.getExternalEntityInfo("incident");
access.disconnect();
return ai.getMappedAttributeNamees();
}

public static void showAvailableSettings
( String username
, String password
, String server
) {
AppExternalAccess access =
    new AppExternalAccess(username, password, server);

// get all Import Mappings
String[] settings= access.getAvailableLoadSettingNames();
Hashtable hSets = new Hashtable();
int iSet = settings.length;

// get items per import mapping
for (int i = 0; i < iSet; i++) {
    access.setSettings(settings[i]);
    Hashtable hItems = new Hashtable();
    String[] items = access.getMappedEntityNames();
    int iLen = items.length;
    // get attributes per item
    // add attributes to inner hashtable
    for (int j = 0; j < iLen; j++) {
        String[] attribs =
            access.getMappedAttributeNamees(items[j]);
        hItems.put(items[j], attribs);
    }
    // add items to outer hashtable
    hSets.put(settings[i], hItems);
}
access.disconnect();
```

```
// print hashtables
StringBuffer sOut = new StringBuffer();
Enumeration eIKeys = hSets.keys();
Enumeration eIVals = hSets.elements();
while (eIKeys.hasMoreElements()) {
    String sSet = (String) eIKeys.nextElement();
    sOut.append("Import Mapping " + sSet + "\r\n\r\n");
    Hashtable hItems = (Hashtable) eIVals.nextElement();
    Enumeration eEKeys = hItems.keys();
    Enumeration eEVals = hItems.elements();
    while (eEKeys.hasMoreElements()) {
        String sEnt = (String) eEKeys.nextElement();
        sOut.append("\tMapped Entity: " + sEnt + "\r\n");
        sOut.append("\tAttributes:\r\n");
        String[] aAttrs = (String[]) eEVals.nextElement();
        int iAttr = aAttrs.length;
        for (int k = 0; k < iAttr; k++) {
            sOut.append("\t\t" + aAttrs[k] + "\r\n");
        }
        sOut.append("\r\n");
    }
    sOut.append("\r\n\r\n");
}
System.out.println(sOut.toString());
}
```

Finding the Proper Entity Instantiation

The `Example3` class shows how to obtain a list of Service Desk entities that comply to a given search criteria. This is important when checking whether a given entity exists, or when retrieving a list of entities with certain characteristics.

In this example, three new classes are introduced. The most important one is the `AppCriterium` class, used to pass search criteria to the method that performs the actual search. In the construction of an `AppCriterium` object, the `AppWhereOperatorEnum` class is used to indicate the logical operators to use when combining `AppCriterium` objects. Apart from this, the `AppAttributeSelection` class is introduced. This class is used to specify the attributes that should be exposed on an entity. These classes belong to the Service Desk IFC and reside in the `com.hp.ifc.util.marshall` package. This package is imported on line 4 of the `Example3` class.

Both methods used in this example produce an array of long values that represent the relevant object IDs. Relevance is determined here by the search criteria defined in the method. Also, the Service Desk entity must be defined for a search action.

An illustration of this process is given in the `findIncidentByLabel()` method. As the name indicates, this method uses reference by label. The construction of the actual list of object IDs is done by the `listMappedEntitiesAsLong()` method defined in the `AppExternalAccess` class (line 58). This method uses two arguments:

- The label for the Service Desk entity within the current import mapping settings.
- An array of `AppCriterium` objects defining the search criteria.

The single `AppCriterium` object used in this example is created in a call to another `AppExternalAccess` method, `createEqualSearchCondition()` on line 49. This is a utility method for the creation of a simple search condition. As arguments it takes:

- an indicator for the logical operator used in the combination of subsequent `AppCriterium` objects. This is a value from `AppWhereOperatorEnum` and can be either `crtAnd` (value 1) or `crtOr` (value 2). The first `AppCriterium` object in an array is

normally given the `and` operator;

- the label for the Service Desk entity;
- the label for the Service Desk attribute involved;
- the attribute value to use in the search. This must always be a `String` object when working with reference by label.

The `findIncidentByNumber ()` method shows the same operation performed using reference by number. The actual list is constructed by the `find ()` method in line 32. The Service Desk attribute is indicated by the use of the `ITSMExternalEnum` class, in this case. In addition to the `AppCriterion` objects, the `AppAttributeSelection` object created on line 19, is passed to the `find ()` method. This object serves to explicitly specify the attributes to retrieve. When only building a list of object oids, no attributes are necessary since the object ID is always retrieved by the API. It is sufficient to use the default selection.

NOTE

The Service Desk application server automatically adds the object ID, and `lockseq-value` attributes used for Service Desk's internal administration of objects retrieved and their status (that is, whether they are updated or not, and if changes have taken place since retrieval or not) when returning a selection. The workflow layer also adds attributes necessary for performing business rules. This prevents failures during the execution of business rules.

The creation of the `AppCriterion` object (line 22) also uses reference by number, diminishing the number of arguments to three. Though the actual search value is a `String` object, it is important to know that the class for the object passed here must be the same as the class defined for the Service Desk attribute referenced.

A number of methods are available in the `AppExternalAccess` class for both ways of listing Service Desk attributes. The methods differ in the arguments they take: long values, `AppOID` objects, `String` objects, and the results they return: arrays of long values, or `AppOID` objects.

The number of methods available for the creation of `AppCriterion` objects is even more abundant. Five different ways exist for passing arguments into methods. Three different types of search criteria that can be used are described below:

- Equality searches

Finding the Proper Entity Instantiation

- Range searches
- Free format searches

Equality and range searches are made accessible with `createEqualSearchCondition()` and `createRangeSearchCondition()` methods. Refer to “Javadoc” on page 61 for more information.

The free-format search methods are more complicated. Again an `AppCriterium` object is being constructed, but some additional arguments can be used. Two extensions make these methods more flexible; the addition of an operator for the interpretation of search values, and the use of a boolean to obtain the negation of a search condition. Together the free-format search methods offer much greater freedom in the construction of tailor-made search conditions.

As for the interpretation of the values specified, this is directed by labels defined in the `com.hp.ifc.util.marshall.AppCriteriumOperatorEnum` class. The available values are listed in the following table. An additional boolean argument can be used to reverse the selection condition. For example; instead of objects that have a value equivalent to yesterday in an attribute, one can specify those that have any value except yesterday. Note that some of these negations are already provided by labels:

Table 3-1 **AppCriteriumOperationEnum Values**

Label	Int value	# values
opEqual	0	1
opNotEqual	1	1
opGreaterThan	2	1
opLessThan	3	1
OpGreaterThanOr EqualTo	4	1
opLessThanOrEqu alTo	5	1
OpBetween	6	2
opNotBetween	7	2
opContains	8	1

Table 3-1 AppCriteriumOperationEnum Values

Label	Int value	# values
opNotContains	9	1
opEmpty	11	0
opNotEmpty	12	0
opYesterday	14	0
opToday	15	0
opTomorrow	16	0
opLast7Days	17	0
opNext7Days	18	0
opLastWeek	19	0
opThisWeek	20	0
opNextWeek	21	0
opLastMonth	22	0
opThisMonth	23	0
opNextMonth	24	0
opStartsWith	25	1

Example 3-3 Finding the Proper Entity Instantiation

```
package com.hp.ifc.ext.examples;

import com.hp.ifc.ext.*;
import com.hp.ifc.util.marshall.*;

/**
 * Example: listing Service Desk objects
 */

public class Example3 {

    public static long[] findIncidentByNumber
```

Examples

Finding the Proper Entity Instantiation

```
( String username
, String password
, String server
) {
AppExternalAccess access =
    new AppExternalAccess(username, password, server);

AppAttributeSelection sel =
    new AppAttributeSelection();

AppCriterium[] crits =
    new AppCriterium[]
    { access.createEqualSearchCondition
      ( AppWhereOperatorEnum.crtAnd
      , ITSMExternalEnum.IncidentDefEnum.atDescription
      , "Server 02 booted"
      )
    };

long[] Incs =
    access.find
      (ITSMExternalEnum.IncidentDefEnum.enOid, sel, crits);

access.disconnect();
return Incs;
}

public static long[] findIncidentByLabel
( String username
, String password
, String server
) {
AppExternalAccess access =
    new AppExternalAccess(username, password, server);

access.setSettings("external_event");

AppCriterium[] crits =
    new AppCriterium[]
    { access.createEqualSearchCondition
      ( AppWhereOperatorEnum.crtAnd
      , "incident"
      , "description"
      , "Server 02 booted"
      )
    };
};
```



```
long[] Incs =  
    access.listMappedEntitiesAsLong("incident", crits);  
  
access.disconnect();  
return Incs;  
}
```

Getting an Entity Instantiation

Example4 shows how to get a Service Desk object into the API programming environment. A large part of this example is copied from the `example3` class, which begins the process for retrieving Service Desk objects.

The `getIncidentByNumber()` method shows the retrieval of a Service Desk incident, using reference by number. This method is similar to the `findIncidentByNumber()` method in `example3`, with a minor extension. After obtaining the list of relevant object IDs, the Service Desk object for each value is retrieved and wrapped in an `AppExternalEntity` object. This is done in line 39:

```
Ents[iter]=access.open(ITSMEExternalEnum.IncidentDefEnum.enOid,  
sel, Incs[i]);
```

Arguments for this method are:

- An identifier for the Service Desk entity.
- An `AppAttributeSelection` object (see “Finding the Proper Entity Instantiation” on page 39).
- The object ID for the Service Desk object.

The return type for this method is the `AppExternalEntity` class. This is the API class for manipulating Service Desk objects using reference by number.

The `getIncidentByLabel()` method shows how to retrieve a Service Desk incident using labels. This is again very much like the `findIncidentByLabel()` method in `Example3`. The essential operation is performed on line 72:

```
Ents[i]=access.openMappedEntity("incident", Incs[i]);
```

This is a simplified variation of the `open()` method explained earlier. This method does not take an `AppAttributeSelection` object as an argument, because the API always retrieves all attributes that are provided with a label, when working with reference by label.

The `openMappedEntity()` method returns an `AppMappedExternalEntity` object. This is an extension of the `AppExternalEntity` object, aimed specifically at the manipulation of Service Desk objects using reference by label.

Example 3-4 Getting an Entity Instantiation

```
package com.hp.ifc.ext.examples;

import com.hp.ifc.ext.*;
import com.hp.ifc.util.marshall.*;

/**
 * Example: retrieval of Service Desk objects
 */

public class Example4 {

    public static AppExternalEntity[] getIncidentByNumber
        ( String username
        , String password
        , String server
        ) {
        AppExternalAccess access =
            new AppExternalAccess(username, password, server);
        AppAttributeSelection sel =
            new AppAttributeSelection();
        sel.putValue
            (ITSMExternalEnum.IncidentDefEnum.atDescription);

        AppCriterium[] crits =
            new AppCriterium[]
            { access.createEqualSearchCondition
              ( AppWhereOperatorEnum.crtAnd
                , ITSMExternalEnum.IncidentDefEnum.atDescription
                , "Server 02 booted"
              )
            };
        long[] Incs =
            access.find
            (ITSMExternalEnum.IncidentDefEnum.enOid, sel, crits);

        AppExternalEntity[] Ents =
            new AppExternalEntity[Incs.length];
        for (int i = 0; i < Incs.length; i++) {
            Ents[i] =
                access.open
                ( ITSMExternalEnum.IncidentDefEnum.enOid
                  , sel
                  , Incs[i]
                );
        }
    }
}
```

Examples

Getting an Entity Instantiation

```
    }
    access.disconnect();
    return Ents;
}

public static AppMappedExternalEntity[] getIncidentByLabel
( String username
, String password
, String server
) {
    AppExternalAccess access =
        new AppExternalAccess(username, password, server);
    access.setSettings("external_event");

    AppCriterium[] crits =
        new AppCriterium[]
        { access.createEqualSearchCondition
          ( AppWhereOperatorEnum.crtAnd
            , "incident"
            , "description"
            , "Server 02 booted"
          )
        };
    long[] Incs =
        access.listMappedEntitiesAsLong("incident", crits);

    AppMappedExternalEntity[] Ents =
        new AppMappedExternalEntity[Incs.length];
    for (int i = 0; i < Incs.length; i++) {
        Ents[i] =
            access.openMappedEntity("incident", Incs[i]);
    }
    access.disconnect();
    return Ents;
}
```

Creating New Objects

Example5 shows how new Service Desk objects are created by means of the `AppExternalAccess` class.

The `makeIncidentByNumber()` method shows how to create Service Desk objects using reference by number. A quick glance at the actual method call on `create()` (line 24), makes clear that it is used in much the same way as the `open()` method used in Example4. The only difference is that an object ID is not passed into this method. The `create()` method returns an `AppExternalEntity` object.

The same applies to the `createMappedEntity()` method used in `makeIncidentByNumber()`, showing reference by label. The `createMappedEntity()` method called on line 41, compares to the `openMappedEntity()` as does the `create()` method to the `open()` method. The `createMappedEntity()` method returns an `AppMappedExternalEntity` object.

Example 3-5 Creating New Objects

```
package com.hp.ifc.ext.examples;

import com.hp.ifc.ext.*;
import com.hp.ifc.util.marshall.*;

/**
 * Example: creating of Service Desk objects
 */

public class Example5 {

    public static AppExternalEntity makeIncidentByNumber
    ( String username
    , String password
    , String server
    ) {
        try {
            AppExternalAccess access =
                new AppExternalAccess(username, password, server);
            AppAttributeSelection sel =
                new AppAttributeSelection();
            sel.putValue
                (ITSMExternalEnum.IncidentDefEnum.atDescription);
```

Examples

Creating New Objects

```
        return
            access.create
                (ITSMEExternalEnum.IncidentDefEnum.enOid, sel);
    }
    catch (ExternalException exc) {
        System.out.println(exc.getMessage());
        return null;
    }
}

public static AppMappedExternalEntity makeIncidentByLabel
( String username
, String password
, String server
) {
    try {
        AppExternalAccess access =
            new AppExternalAccess(username, password, server);
        access.setSettings("external_event");
        return access.createMappedEntity("incident");
    }
    catch (ExternalException exc) {
        System.out.println(exc.getMessage());
        return null;
    }
}
```

Getting and Setting Attribute Values

The `Example6` class shows the creation of a new Service Desk object, and the elementary manipulation of one of the object's attributes.

The first example method, `changeIncidentByNumber()`, as always applies to reference by number. After creating a new incident object, this method shows the contents of the description attribute. This will be empty (null) since we are working with a new incident.

On line 33, a value is assigned to the description. This is done by calling the `setValue()` method on the `AppExternalEntity` object that represents the incident. The `changeIncidentByNumber()` shows the attribute's contents, that are now set to "An example".

In this example a `String` object is passed into `setValue()`, this method expects to receive an appropriate object type. The Java types you are most likely to encounter are listed in Table 3-2. You will need to determine the type of Service Desk attribute to be manipulated in order to use the table. The `determineAttributeType()` method can be used to determine this, when needed.

The `changeIncidentByLabel()` method shows the same sequence of events using reference by label. Apart from the differences in object creation that are already familiar from `Example5`, it will appear that the description attribute already has a value the first time it is shown. This is caused by the application of default values from the template, assigned to the incident entity defined in the import settings.

Setting the attribute value on line 66 differs little from the approach used in `changeIncidentByNumber()`. Apart from using reference by label the `setValue()` method defined for `AppMappedExternalEntity` always takes a `String` object for the value.

Table 3-2

Java Types

Service Desk Attribute Type	Java Data Type
Boolean (Yes/No)	Boolean
Currency (money)	Double

Getting and Setting Attribute Values**Table 3-2****Java Types**

Service Desk Attribute Type	Java Data Type
Date (without time)	Com.ms.wfc.app.Time or Double representing the number of hundred nanosecond units elapsed since January 1, 100AD 12:00 midnight.
Datetime (timestamp)	Com.ms.wfc.app.Time or Double representing the number of hundred nanosecond units elapsed since January 1, 100AD 12:00 midnight.
Description (length 80)	String
Double	Double
Duration	Double representing number of minutes.
Email	String
EntityReference	AppOID
Gender	AppOID (0=male, 1=female)
Integer	Integer
Long	Long
Longtext (memo)	String
Name (length 50)	String
Searchcode (length 50)	String (Searchcode must be all capitals, cannot contain any spaces or the characters: ':', '*', '_', '%', and cannot start with any numeric characters.
Shorttext (length 40)	String
Telephone	String
Text (length 225)	String
Text64kB	String

Example 3-6 Getting and Setting Attribute Values

```
package com.hp.ifc.ext.examples;

import com.hp.ifc.ext.*;
import com.hp.ifc.rep.*;
import com.hp.ifc.types.*;
import com.hp.ifc.util.marshall.*;

/**
 * Example: manipulating Service Desk objects
 */

public class Example6 {

    public static void changeIncidentByNumber
    ( String username
    , String password
    , String server
    ) {
        Object s;
        try {
            AppExternalAccess access =
                new AppExternalAccess(username, password, server);
            AppAttributeSelection sel =
                new AppAttributeSelection();
            sel.putValue
                (ITSMExternalEnum.IncidentDefEnum.atDescription);
            AppExternalEntity oEnt =
                access.create
                    (ITSMExternalEnum.IncidentDefEnum.enOid, sel);

            s = oEnt.getValue
                (ITSMExternalEnum.IncidentDefEnum.atDescription);
            System.out.println("Description is " + s);
            oEnt.setValue
                ( ITSMExternalEnum.IncidentDefEnum.atDescription
                , "An example"
                );
            s = oEnt.getValue
                (ITSMExternalEnum.IncidentDefEnum.atDescription);
            System.out.println("Description is set to " + s);
            access.disconnect();
        }
        catch (ExternalException exc) {
            System.out.println(exc.getMessage());
        }
    }
}
```

Examples

Getting and Setting Attribute Values

```
    }
}

public static String determineAttributeType(long attId) {
    AppAttributeInfo ai =
        AppObjectModel.getAttributeInfo(new AppOID(attId));
    return ai.getAttributeTypeInfo().getName();
}

public static void changeIncidentByLabel
( String username
, String password
, String server
) {
    Object s;
    try {
        AppExternalAccess access =
            new AppExternalAccess(username, password, server);
        access.setSettings("external_event");
        AppMappedExternalEntity oMEnt =
            access.createMappedEntity("incident");
        s=oMEnt.getValue("description");
        System.out.println("Description is " + s);
        oMEnt.setValue("description", "Another example");
        s=oMEnt.getValue("description");
        System.out.println("Description is set to " + s);
        access.disconnect();
    }
    catch (ExternalException exc) {
        System.out.println(exc.getMessage());
    }
}
```

Saving Instantiations

Example7 shows how to save an object to the Service Desk environment. In doing so, the new or changed object is made available for use by other Service Desk users. This example in fact is a minor extension of the previous example.

In both methods of the Example7 class the object is stored by a call to the save() method on the AppExternalEntity class. In saveIncidentByLabel(), the AppMappedExternalEntity class is used, but the save() method is inherited from AppExternalEntity. The calls can be found on lines 45 and 64. This method saves the Service Desk object wrapped in the AppExternalEntity object to the Service Desk database.

The saveIncidentByNumber() method can produce an error message. Since only some attributes are set, the Service Desk application server can generate an exception with the text “you must fill in the <attribute> box”. This demonstrates that it is important to know which attributes must be filled, because if these are not set, a new object will not be saved.

The saveIncidentByLabel() method doesn't report an error, because the mandatory attributes are filled from the template defined by the import settings. This demonstrates a big advantage of working with the import settings: it is possible to define default values that make sense. One can even define several sets for a single Service Desk entity, that will then be applied for different labels. Of course, using a template will only prevent messages caused by empty attributes when all mandatory attributes have received a template value.

The AppExternalEntity class has two methods that are more or less analogous to the save() method:

- delete()
This method will remove the Service Desk object being processed (as far as business rules allow this). This clearly does not work on newly created objects.
- rollback()
This method will revert all changes on the Service Desk object being processed, since it was created or retrieved from Service Desk. If rollback() was applied previously, rollback() will revert all

Examples

Saving Instantiations

changes applied since the previous call on `rollback()`.

Example 3-7 Saving Instantiations

```
package com.hp.ifc.ext.examples;

import com.hp.ifc.ext.*;
import com.hp.ifc.util.marshall.*;

/**
 * Example: manipulating and saving Service Desk objects
 */

public class Example7 {

    public static void saveIncidentByNumber
        ( String username
        , String password
        , String server
        ) {
        try {
            AppExternalAccess access =
                new AppExternalAccess(username, password, server);
            AppAttributeSelection sel =
                new AppAttributeSelection();
            AppAttributeSelection subSel =
                new AppAttributeSelection();
            sel.putValue
                (ITSMEExternalEnum.IncidentDefEnum.atDescription);
            sel.putValue
                (ITSMEExternalEnum.IncidentDefEnum.atInformation);
            sel.putValue
                (ITSMEExternalEnum.IncidentDefEnum.atStatus, subSel);

            AppExternalEntity oEnt =
                access.create
                    (ITSMEExternalEnum.IncidentDefEnum.enOid, sel);
            oEnt.setValue
                ( ITSMEExternalEnum.IncidentDefEnum.atDescription
                , "An example"
                );
            oEnt.setValue
                ( ITSMEExternalEnum.IncidentDefEnum.atInformation
                , "An example"
                );
            oEnt.setValue
```

```
        ( ITSMExternalEnum.IncidentDefEnum.atStatus
          , "Registered"
        );
    oEnt.save();
    access.disconnect();
    System.out.println("New incident is saved");
}
catch (ExternalException exc) {
    System.out.println(exc.getMessage());
}
}

public static void saveIncidentByLabel
( String username
, String password
, String server
) {
    try {
        AppExternalAccess access =
            new AppExternalAccess(username, password, server);
        access.setSettings("external_event");
        AppMappedExternalEntity oEnt =
            access.createMappedEntity("incident");
        oEnt.setValue("description", "Another example");
        oEnt.save();
        access.disconnect();
        System.out.println("New incident is saved");
    }
    catch (ExternalException exc) {
        System.out.println(exc.getMessage());
    }
}
}
```

Single Instantiation Processing

Example8 shows a simplified way of working with the Service Desk API. By using two utility classes you simplify the tasks involved in opening, creating, and saving `App(Mapped)ExternalEntities` and setting their values. A single method call can create a Service Desk object, or a relation between two Service Desk objects. The one condition is that import settings must be defined for these objects, because the methods use reference by label.

The `relateObjects()` method shows the creation of two Service Desk objects and a relation between these two objects. The actual objects are created using the `AppSingleLoad.processEntity()` method (lines 32 and 37), the relation is created by the `AppRelationLoad.processRelation()` method (line 50).

To get into the `AppSingleLoad` class first, this has several overloaded versions of the `processEntity()` method. The one used here is the simplest, that takes four arguments:

- The label for the Service Desk entity within the current import mapping.
- An array of labels for the Service Desk attributes whose values are to be set.
This is created, and filled on line 30. It is the API programmer's responsibility to include all attribute labels that are defined as key attributes for the entity, otherwise an exception will be thrown.
- An array of `String` objects, specifying the values to which these attributes will be set.
This second array is created and filled on line 31. Labels and values are combined according to their positions in the arrays.
- A boolean value, indicating whether the Service Desk object defined must be saved (false) or removed (true).

A number of prominent Service Desk entities are provided with a `Source ID`. This attribute is intended for the storage of object identifiers from external systems. This is illustrated here by the use of the `NNM_ID` label, this is used to store an external `NNM ID` field in the `Source ID` attribute. This use of the `Source ID` field is strongly recommended.

Lines 35 through 36 show the re-use of previously defined Java objects to

process a second configuration item.

The `AppSingleLoad` class has a number of `processEntity()` methods. The ones not shown here take the following additional arguments:

- The import setting name, relieving us from the need to set that attribute (can be important when different import settings are used in one program).
- The import setting name, plus a Service Desk user name, password, and server.

The `AppRelationLoad` class has just one method, `processRelation()` (lines 50, 51). This method takes five arguments:

- The label for the first Service Desk entity.
- An array with key attributes and values, as created on line 42 through 44. These are used for identifying the first Service Desk object.
- A label describing the kind of Service Desk relation.
- The label for the second Service Desk entity.
- An array with key attributes and their values, as created on line 46 through 48. These are used for the identification of the second Service Desk object.

It is the API programmer's responsibility to include all key attributes. If all key attributes are not included, an exception will be thrown, because the object cannot be identified.

Example 3-8 Single Instantiation Processing

```
package com.hp.ifc.ext.examples;

import com.hp.ifc.ext.*;
import com.hp.ifc.util.marshal.*;

/**
 * Example: simplified manipulation of objects and relations
 */

public class Example8 {

    public static void relateObjects
        ( String username
        , String password
```

Examples

Single Instantiation Processing

```
, String server
) {
try {
    String nextNumber = "0008";
    String segmentId = "NNM_Segment_" + nextNumber;
    String segmentName = "Test_Segment_" + nextNumber;
    String networkId = "NNM_Network_" + nextNumber;
    String networkName = "Test_Network_" + nextNumber;
    String[] aAttr, aVals;

    AppExternalAccess access =
        new AppExternalAccess(username, password, server);
    access.setSettings("nnm6_import");

    AppSingleLoad load = new AppSingleLoad(access);

    aAttr = new String[] {"NNM_ID", "NAME"};
    aVals = new String[] {segmentId, segmentName};
    load.processEntity("SEGMENT", aAttr, aVals, false);
    System.out.println("Saved segment " + segmentId);

    aAttr = new String[] {"NNM_ID", "NAME"};
    aVals = new String[] {networkId, networkName};
    load.processEntity("NETWORK", aAttr, aVals, false);
    System.out.println("Saved network " + networkId);

    AppRelationLoad relate = new AppRelationLoad(access);

    String[][] aParentKey = new String[2][1];
    aParentKey[0][0] = "NNM_ID";
    aParentKey[1][0] = segmentId;

    String[][] aChildKey = new String[2][1];
    aChildKey[0][0] = "NNM_ID";
    aChildKey[1][0] = networkId;

    relate.processRelation
        ( "SEGMENT", aParentKey, "Parent"
        , "NETWORK", aChildKey
        );
    System.out.println
        ("Saved relation: " + segmentId + " - " + networkId);
    access.disconnect();
}
catch (ExternalException exc) {
    System.out.println(exc.getMessage());
}
```



```
}  
}
```

Error Presentation

Example9 demonstrates how Java exceptions, thrown by the Service Desk API classes, can be caught and displayed. This has already been shown in a number of other classes, but the Example9 class is more detailed.

The Service Desk application server normally throws a `ComFailException` when something unexpected happens. Since these exceptions are Microsoft-specific, the Service Desk API throws them again as `ExternalExceptions`. These are generic `Exception` objects, that also have a severity attribute. This is used to distinguish between errors, warnings and information messages. The severity values are defined in `com.hp.ifc.rep.AppSeverityEnum`, as imported in Example9 on line 5.

The `AppExternalAccess` class has a method `getMessage()` that re-throws any exception passed to it as an `ExternalException`. Any `Exception` caught by the API classes is re-thrown by this method, so these are all caught with an exception handler on `ExternalExceptions`. If the exception passed is a `ComFailException`, its severity will be taken over. All other exceptions will be assigned severity `svCritical`. Since other kinds of exceptions can always be thrown, it is a good idea to catch all exceptions, this example doesn't do this.

The Example9 class contains an exception handler on both member methods. This shows how context-specific error messages are generated from an `ExternalException` (lines 31-37 and 56-61 using the `switch` statement). The exception handlers record the `ExternalException`'s severity (lines 32 and 57), and use this to determine a prefix for the error message.

This example can be expanded using different ways of displaying or logging the external exceptions thrown. In doing so, error reporting can be brought in line with the API programmer's organizational standards.

Example 3-9

Error Presentation

```
package com.hp.ifc.ext.examples;

import com.hp.ifc.ext.*;
import com.hp.ifc.util.marshal.*;
import com.hp.ifc.rep.AppSeverityEnum;
```

```
/**
 * Example: extended error reporting
 */

public class Example9 {

    public static void saveIncidentByNumber
    ( String username
    , String password
    , String server
    ) {
    try {
        AppExternalAccess access =
            new AppExternalAccess(username, password, server);
        AppAttributeSelection sel =
            new AppAttributeSelection();
        sel.putValue
            (ITSMExternalEnum.IncidentDefEnum.atDescription);
        AppExternalEntity oEnt =
            access.create
                (ITSMExternalEnum.IncidentDefEnum.enOid, sel);
        oEnt.save();
        access.disconnect();
        System.out.println("New incident is saved");
    }
    catch (ExternalException exc) {
        String s = "INFO";
        switch(exc.getSeverity()) {
            case AppSeverityEnum.svCritical: s = "ERROR"; break;
            case AppSeverityEnum.svWarning: s = "WARNING"; break;
        }
        System.out.println(s + ": " + exc.getMessage());
    }
    }

    public static void saveIncidentByLabel
    ( String username
    , String password
    , String server
    ) {
    try {
        AppExternalAccess access =
            new AppExternalAccess(username, password, server);
        access.setSettings("external_event");
        AppMappedExternalEntity oMEnt =
```

Examples

Error Presentation

```
        access.createMappedEntity("incident");
        oMEnt.setValue("description", "dummy description");
        oMEnt.setValue("status", null);
        oMEnt.save();
        access.disconnect();
        System.out.println("New incident is saved");
    }
    catch (ExternalException exc) {
        String s = "INFO";
        switch(exc.getSeverity()) {
            case AppSeverityEnum.svCritical: s = "ERROR"; break;
            case AppSeverityEnum.svWarning: s = "WARNING"; break;
        }
        System.out.println(s + ": " + exc.getMessage());
    }
}
```

4 **Javadoc**

The Javadoc document is generated from the API source files. The javadoc describes the classes, interfaces, constructors, methods, and fields in the API.

Javadoc Elements

The javadoc contains the following classes and other elements:
AppExternalAccess, AppExternalEntity, AppMappedExternalEntity,
AppExternalUtilities, ExternalException, AppSingleLoad,
AppRelationLoad.

The information in the following sections is also available in HTML format. The HTML javadoc also includes information about the ITSMExternalEnum class, which is not listed in this guide. You can find the HTML javadoc on the HP OpenView Service Desk 3.0 CD-ROM, in the \\Doc\API Javadoc folder. To open the javadoc, open the index.html file. This file contains links to the other files in the \\Doc\API Javadoc folder.

package com.hp.ifc.ext

Description

Class Summary

Classes

<u>AppExternalAccess</u>	Access for external software to the HP OpenView Service Desk environment.
<u>AppExternalEntity</u>	Access for external software to entities that are part of the HP OpenView Service Desk environment.
<u>AppExternalUtilities</u>	Some utility methods to use within the HP OpenView Service Desk external access context.
<u>AppMappedExternalEntity</u>	Access for external software to entities that are part of the HP OpenView Service Desk environment.
<u>AppRelationLoad</u>	Specialized class to process a single relation (association) to be added to the HP OpenView Service Desk system.
<u>AppSingleLoad</u>	Utility class for processing a single instantiation of an entity that is part of the HP OpenView Service Desk system.

Exceptions

<u>ExternalException</u>	Exception class for use within the HP OpenView Service Desk external access context.
--	--

Javadoc
Package Summary

com.hp.ifc.ext AppExternalAccess

Syntax

```
public final class AppExternalAccess extends java.lang.Object  
  
java.lang.Object  
|  
+--com.hp.ifc.ext.AppExternalAccess
```

Description

Access for external software to the HP OpenView Service Desk environment. This class offers general services relevant for external access. These are:

- Connecting to a newly started `AppWorkflowManager` and logging in
- Disconnect from `AppWorkflowManager`
- Setting a definition for the integration's settings
- Presenting a list of the mapped entities
- Casting `Exceptions` thrown from the Service Desk environment to `ExternalExceptions`
- Creating the `AppExternalEntity` objects used in the integration (these objects persists themselves)
- Creating search criteria for the identification of Service Desk entities

In general, this kind of integration uses the settings concept to find out what is allowed and what is not, and what settings apply to a given integration.

See Also: `com.hp.ifc.rep.ext.AppExternalDataModel`,
`com.hp.ifc.wf.AppWorkflowManager`, [AppExternalEntity](#)

Member Summary

Fields

[REVISION](#)

Constructors

Member Summary

<code>AppExternalAccess(AppSession)</code>	Connects into the Service Desk environment using an already-opened session.
<code>AppExternalAccess(String, String, String)</code>	Connects into the Service Desk environment, using the parameters as authentication.
Methods	
<code>create(AppOID, AppAttributeSelection)</code>	Creates an <code>AppExternalEntity</code> object for a given external entity, exposing the attributes specified in the <code>AppAttributeSelection</code> object passed.
<code>create(long, AppAttributeSelection)</code>	Creates an <code>AppExternalEntity</code> object for a given external entity, exposing the attributes specified in the <code>AppAttributeSelection</code> object passed.
<code>createEqualSearchCondition(int, AppOID[], Object)</code>	Creates a search condition (<code>AppCriterion</code> object) that specifies an equality search on an attribute belonging to a related entity.
<code>createEqualSearchCondition(int, AppOID, Object)</code>	Creates a search condition (<code>AppCriterion</code> object) that specifies a simple equality search.
<code>createEqualSearchCondition(int, long[], Object)</code>	Creates a search condition (<code>AppCriterion</code> object) that specifies an equality search on an attribute belonging to a related entity.
<code>createEqualSearchCondition(int, long, Object)</code>	Creates a search condition (<code>AppCriterion</code> object) that specifies a simple equality search.
<code>createEqualSearchCondition(int, String, String, Object)</code>	Creates a search condition (<code>AppCriterion</code> object) that specifies a simple equality search.
<code>createMappedEntity(AppOID)</code>	Creates an <code>AppMappedExternalEntity</code> object for a given mapped external entity.
<code>createMappedEntity(long)</code>	Creates an <code>AppMappedExternalEntity</code> object for a given mapped external entity.
<code>createMappedEntity(String)</code>	Creates an <code>AppMappedExternalEntity</code> object for a given mapped external entity.
<code>createRangeSearchCondition(int, AppOID[], Object, Object)</code>	Creates a search condition (<code>AppCriterion</code> object) that specifies a range search on an attribute belonging to a related entity.
<code>createRangeSearchCondition(int, AppOID, Object, Object)</code>	Creates a search condition (<code>AppCriterion</code> object) that specifies a range search.
<code>createRangeSearchCondition(int, long[], Object, Object)</code>	Creates a search condition (<code>AppCriterion</code> object) that specifies a range search on an attribute belonging to a related entity.

Member Summary

<code>createRangeSearchCondition(int, long, Object, Object)</code>	Creates a search condition (AppCriterion object) that specifies a range search.
<code>createRangeSearchCondition(int, String, String, Object, Object)</code>	Creates a search condition (AppCriterion object) that specifies a range search.
<code>createSearchCondition(int, int, AppOID[], Object, boolean)</code>	Creates a tailor-made search condition (AppCriterion object).
<code>createSearchCondition(int, int, long[], Object, Object, boolean)</code>	Creates a tailor-made search condition (AppCriterion object).
<code>createSearchCondition(int, int, String, String, Object, Object, boolean)</code>	Creates a tailor-made search condition (AppCriterion object).
<code>disconnect()</code>	Disconnects the client from the server.
<code>find(AppOID, AppAttributeSelection, AppCriterion[])</code>	Returns an array of AppOID objects representing the instances of a given external entity that satisfy the given selection criteria.
<code>find(long, AppAttributeSelection, AppCriterion[])</code>	Returns an array of long values representing the instances of a given external entity that satisfy the given selection criteria.
<code>findMappedEntities(AppOID, AppCriterion[])</code>	Returns an array of AppOID objects representing the instances of a given mapped external entity that satisfy the given selection criteria.
<code>findMappedEntities(long, AppCriterion[])</code>	Returns an array of long values representing the instances of a given mapped external entity that satisfy the given selection criteria.
<code>findMappedEntities(String, AppCriterion[])</code>	Returns an array of AppOID objects representing the instances of a given mapped external entity that satisfy the given selection criteria.
<code>findMappedEntitiesAsLong(String, AppCriterion[])</code>	Returns an array of long values representing the instances of a given mapped external entity that satisfy the given selection criteria.
<code>getAvailableLoadSettingNames()</code>	Returns an array of Strings for the names of the available load settings.
<code>getAvailableLoadSettings()</code>	Returns an array of AppOID objects representing the available load settings.
<code>getAvailableLoadSettingsAsLong()</code>	Returns an array of long values representing the available load settings.

Member Summary

<code>getExternalEntityInfo(AppOID)</code>	Returns the <code>AppExternalEntityInfo</code> object specified by the <code>AppOID</code> object.
<code>getExternalEntityInfo(long)</code>	Returns the <code>AppExternalEntityInfo</code> object specified by the <code>long</code> value.
<code>getExternalEntityInfo(String)</code>	Returns the <code>AppExternalEntityInfo</code> object specified by the name.
<code>getMappedAttributeAsLong(long)</code>	Returns an array of <code>long</code> values representing the attributes defined for a given external entity (<code>AppExternalAttributeInfo</code> objects).
<code>getMappedAttributeNames(String)</code>	Returns an array of <code>String</code> objects for the attribute names defined for a given external entity (<code>AppExternalAttributeInfo</code> objects).
<code>getMappedAttributes(AppOID)</code>	Returns an array of <code>AppOID</code> objects representing the attributes defined for a given external entity (<code>AppExternalAttributeInfo</code> objects).
<code>getMappedEntities()</code>	Returns an array of <code>AppOID</code> objects representing the entities defined in the active import settings (<code>AppExternalEntityInfo</code> objects).
<code>getMappedEntitiesAsLong()</code>	Returns an array of <code>long</code> values representing the entities defined in the active import settings (<code>AppExternalEntityInfo</code> objects).
<code>getMappedEntityNames()</code>	Returns an array of <code>Strings</code> for the names of the entities defined in the active import settings (<code>AppExternalEntityInfo</code> objects).
<code>getMessage(Exception)</code>	Re-throws an <code>Exception</code> object as an <code>ExternalException</code> object.
<code>listMappedEntities(AppOID, AppCriterion[])</code>	Returns an array of <code>AppOID</code> objects representing the instances of a given mapped external entity that satisfy the given selection criteria.
<code>listMappedEntities(long, AppCriterion[])</code>	Returns an array of <code>long</code> values representing the instances of a given mapped external entity that satisfy the given selection criteria.
<code>listMappedEntities(String, AppCriterion[])</code>	Returns an array of <code>AppOID</code> objects representing the instances of a given mapped external entity that satisfy the given selection criteria.
<code>listMappedEntitiesAsLong(String, AppCriterion[])</code>	Returns an array of <code>long</code> values representing the instances of a given mapped external entity that satisfy the given selection criteria.
<code>loggedIn()</code>	Returns <code>true</code> when connected into the Service Desk environment, otherwise <code>false</code> .
<code>open(AppOID, AppAttributeSelection, AppOID)</code>	Creates an <code>AppExternalEntity</code> object for a given external entity and fills this with data on a given Service Desk entity instantiation.
<code>open(long, AppAttributeSelection, long)</code>	Creates an <code>AppExternalEntity</code> object for a given external entity and fills this with data on a given Service Desk entity instantiation.
<code>openMappedEntity(AppOID, AppOID)</code>	Creates an <code>AppMappedExternalEntity</code> object for a given mapped external entity and fills this with data on a given Service Desk entity instantiation.

Member Summary

<code><u>openMappedEntity(long, long)</u></code>	Creates an <code>AppMappedExternalEntity</code> object for a given mapped external entity and fills this with data on a given Service Desk entity instantiation.
<code><u>openMappedEntity(String, long)</u></code>	Creates an <code>AppMappedExternalEntity</code> object for a given mapped external entity and fills this with data on a given Service Desk entity instantiation.
<code><u>reconnect()</u></code>	Reconnects the client to the server.
<code><u>setSettings(AppOID)</u></code>	Sets the given set of import settings as the active set.
<code><u>setSettings(long)</u></code>	Sets the given set of import settings as the active set.
<code><u>setSettings(String)</u></code>	Sets the given set of import settings as the active set.
<code><u>shutdown()</u></code>	Disconnects and shuts down the program.

Inherited Member Summary

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`

Fields

REVISION

```
public static final java.lang.String REVISION
```

Constructors

`AppExternalAccess(AppSession)`

```
public AppExternalAccess(com.hp.ifc.wf.AppSession session)
```

AppExternalAccess

Connects into the Service Desk environment using an already-opened session. This `AppSession` object is obtained from the `AppWorkflowManager`.

Parameters:

`session` - the Service Desk session used

See Also: `com.hp.ifc.wf.AppSession`,
`com.hp.ifc.wf.AppWorkflowManager`

AppExternalAccess(String, String, String)

```
public AppExternalAccess(java.lang.String accountName,  
                          java.lang.String password, java.lang.String  
                          appServer)
```

Connects into the Service Desk environment, using the parameters as authentication. This method starts a new `AppWorkflowManager` object, and when run on the Service Desk application server the other layers as well.

Parameters:

`accountName` - the logical account name as used in the Service Desk environment

`password` - the password used to authenticate the Service Desk account

`appServer` - the name or IP address of the application server

Throws: `ExternalException` - when login fails

See Also: `com.hp.ifc.wf.WorkflowManager`

Methods

create(AppOID, AppAttributeSelection)

```
public AppExternalEntity create(  
    com.hp.ifc.types.AppOID entOID,  
    com.hp.ifc.util.marshall.AppAttributeSelection sel)
```

Creates an `AppExternalEntity` object for a given external entity, exposing the attributes specified in the `AppAttributeSelection` object passed.

The new `AppExternalEntity` object can be used to insert a new instance of the given entity into the Service Desk environment.

Use this method after successfully logging on.

Parameters:

`entOid` - `AppOID` object for an external entity

`sel` - `AppAttributeSelection` object specifying the attributes to expose

Returns: `AppExternalEntity` object for the given external entity

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: `AppMappedExternalEntity`, `#setSettings()`, `#createAttribSelection()`

create(long, AppAttributeSelection)

```
public AppExternalEntity create(long entity,
                               com.hp.ifc.util.marshall.AppAttributeSelection sel)
```

Creates an `AppExternalEntity` object for a given external entity, exposing the attributes specified in the `AppAttributeSelection` object passed.

The new `AppExternalEntity` object can be used to insert a new instance of the given entity into the Service Desk environment.

This method to execute after succesful login on the Service Desk system.

Parameters:

`entity` - long value for an external entity

Returns: `AppExternalEntity` object for the given external entity

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: `AppExternalEntity`, `#setSettings()`, `#createAttribSelection()`

AppExternalAccess**createEqualSearchCondition(int, AppOID[], Object)**

```
public com.hp.ifc.util.marshall.AppCriterium  
    createEqualSearchCondition(int operator,  
        com.hp.ifc.types.AppOID[] attrib, java.lang.Object  
        value)
```

Creates a search condition (`AppCriterium` object) that specifies an equality search on an attribute belonging to a related entity.

Parameters:

`operator` - logical operator for the combination of search conditions (value from

`com.hp.ifc.util.marshall.AppWhereOperatorEnum`)

`attrib` - AppOID array for the Service Desk attribute. This array describes the actual attribute and its relation to the main entity

`value` - Object representation of the value to search on

Returns: `AppCriterium` object implementing the search condition defined

See Also: `com.hp.ifc.util.marshall.AppCriterium`

createEqualSearchCondition(int, AppOID, Object)

```
public com.hp.ifc.util.marshall.AppCriterium  
    createEqualSearchCondition(int operator,  
        com.hp.ifc.types.AppOID attrib, java.lang.Object  
        value)
```

Creates a search condition (`AppCriterium` object) that specifies a simple equality search.

Parameters:

`operator` - logical operator for the combination of search conditions (value from

`com.hp.ifc.util.marshall.AppWhereOperatorEnum`)

`attrib` - AppOID object for the Service Desk attribute

`value` - Object representation of the value to search on

Returns: `AppCriterium` object implementing the search condition defined

See Also: `com.hp.ifc.util.marshall.AppCriterium`

createEqualSearchCondition(int, long[], Object)

```
public com.hp.ifc.util.marshal.AppCriterium  
    createEqualSearchCondition(int operator, long[]  
        attrib, java.lang.Object value)
```

Creates a search condition (AppCriterium object) that specifies an equality search on an attribute belonging to a related entity.

Parameters:

`operator` - logical operator for the combination of search conditions (value from `com.hp.ifc.util.marshal.AppWhereOperatorEnum`)

`attrib` - long array for the Service Desk attribute. This array describes the actual attribute and its relation to the main entity (values from `ITSMEExternalEnum`)

`value` - Object representation of the value to search on

Returns: AppCriterium object implementing the search condition defined

See Also: `com.hp.ifc.util.marshal.AppCriterium`

createEqualSearchCondition(int, long, Object)

```
public com.hp.ifc.util.marshal.AppCriterium  
    createEqualSearchCondition(int operator, long  
        attrib, java.lang.Object value)
```

Creates a search condition (AppCriterium object) that specifies a simple equality search.

Parameters:

`operator` - logical operator for the combination of search conditions (value from `com.hp.ifc.util.marshal.AppWhereOperatorEnum`)

`attrib` - long value for the Service Desk attribute (value from `ITSMEExternalEnum`)

`value` - Object representation of the value to search on

Returns: AppCriterium object implementing the search condition defined

See Also: `com.hp.ifc.util.marshal.AppCriterium`

createEqualSearchCondition(int, String, String, Object)

```
public com.hp.ifc.util.marshal.AppCriterium  
    createEqualSearchCondition(int operator,  
        java.lang.String entity, java.lang.String attrib,  
        java.lang.Object value)
```

Creates a search condition (AppCriterium object) that specifies a simple equality search.

This method to execute after setting the active import settings.

Parameters:

operator - logical operator for the combination of search conditions (value from com.hp.ifc.util.marshal.AppWhereOperatorEnum)
entity - String object for a mapped external entity name
attrib - String object for a mapped external attribute name
value - Object representation of the value to search on

Returns: AppCriterium object implementing the search condition defined

See Also: com.hp.ifc.util.marshal.AppCriterium

createMappedEntity(AppOID)

```
public AppMappedExternalEntity  
    createMappedEntity(com.hp.ifc.types.AppOID extEnt)
```

Creates an AppMappedExternalEntity object for a given mapped external entity. The attributes to expose on the AppExternalEntity object are determined from the mapped attributes defined on the external object. The template defined with the mapped entity is applied after creation of the new object, thus effectively setting all default values.

The new AppMappedExternalEntity object can be used to insert a new instance of the given entity into the Service Desk environment.

This method to execute after the setSettings method.

Parameters:

extEnt - AppOID representation of a mapped external entity

Returns: AppMappedExternalEntity object for the given external entity

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: [AppMappedExternalEntity](#), `#setSettings()`, `#createAttribSelection()`

createMappedEntity(long)

```
public AppMappedExternalEntity createMappedEntity(long extEnt)
```

Creates an `AppMappedExternalEntity` object for a given mapped external entity. The attributes to expose on the `AppExternalEntity` object are determined from the mapped attributes defined on the external object. The template defined with the mapped entity is applied after creation of the new object, thus effectively setting all default values.

The new `AppMappedExternalEntity` object can be used to insert a new instance of the given entity into the Service Desk environment.

This method to execute after the `setSettings` method.

Parameters:

`extEnt` - long representation of a mapped external entity

Returns: `AppMappedExternalEntity` object for the given external entity

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: [AppMappedExternalEntity](#), `#setSettings()`, `#createAttribSelection()`

createMappedEntity(String)

```
public AppMappedExternalEntity  
    createMappedEntity(java.lang.String extName)
```

Creates an `AppMappedExternalEntity` object for a given mapped external entity. The attributes to expose on the `AppExternalEntity` object are determined from the mapped attributes defined on the external object. The template defined with the mapped entity is applied after creation of the new object, thus effectively setting all default values.

The new `AppMappedExternalEntity` object can be used to insert a new instance of the given entity into the Service Desk environment.

AppExternalAccess

This method to execute after the `setSettings` method.

Parameters:

`extName` - String object for a mapped external entity name

Returns: `AppExternalEntity` object for the given external entity

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: `AppExternalEntity`, `#setSettings()`,
`#createAttribSelection()`

createRangeSearchCondition(int, AppOID[], Object, Object)

```
public com.hp.ifc.util.marshall.AppCriterium
    createRangeSearchCondition(int operator,
        com.hp.ifc.types.AppOID[] attrib, java.lang.Object
        fromVal, java.lang.Object toVal)
```

Creates a search condition (`AppCriterium` object) that specifies a range search on an attribute belonging to a related entity.

Parameters:

`operator` - logical operator for the combination of search conditions (value from

`com.hp.ifc.util.marshall.AppWhereOperatorEnum`)

`attrib` - `AppOID` array for the Service Desk attribute. This array describes the actual attribute and its relation to the main entity

`fromVal` - `Object` representation of the lowest value for the search range

`toVal` - `Object` representation of the highest value for the search range

Returns: `AppCriterium` object implementing the search condition defined

See Also: `com.hp.ifc.util.marshall.AppCriterium`

createRangeSearchCondition(int, AppOID, Object, Object)

```
public com.hp.ifc.util.marshall.AppCriterium
    createRangeSearchCondition(int operator,
```

```
com.hp.ifc.types.AppOID attrib, java.lang.Object  
fromVal, java.lang.Object toVal)
```

Creates a search condition (AppCriterium object) that specifies a range search.

Parameters:

operator - logical operator for the combination of search conditions (value from

com.hp.ifc.util.marshal.AppWhereOperatorEnum)

attrib - the AppOID object representing the Service Desk attribute

fromVal - Object representation of the lowest value for the search range

toVal - Object representation of the highest value for the search range

Returns: AppCriterium object implementing the search condition defined

See Also: com.hp.ifc.util.marshal.AppCriterium

createRangeSearchCondition(int, long[], Object, Object)

```
public com.hp.ifc.util.marshal.AppCriterium  
createRangeSearchCondition(int operator, long[]  
attrib, java.lang.Object fromVal, java.lang.Object  
toVal)
```

Creates a search condition (AppCriterium object) that specifies a range search on an attribute belonging to a related entity.

Parameters:

operator - logical operator for the combination of search conditions (value from

com.hp.ifc.util.marshal.AppWhereOperatorEnum)

attrib - long array for the Service Desk attribute. This array describes the actual attribute and its relation to the main entity (values from ITSMExternalEnum)

fromVal - Object representation of the lowest value for the search range

AppExternalAccess

toVal - Object representation of the highest value for the search range

Returns: AppCriterium object implementing the search condition defined

See Also: com.hp.ifc.util.marshall.AppCriterium

createRangeSearchCondition(int, long, Object, Object)

```
public com.hp.ifc.util.marshall.AppCriterium
    createRangeSearchCondition(int operator, long
        attrib, java.lang.Object fromVal, java.lang.Object
            toVal)
```

Creates a search condition (AppCriterium object) that specifies a range search.

Parameters:

operator - logical operator for the combination of search conditions (value from

com.hp.ifc.util.marshall.AppWhereOperatorEnum)

attrib - the long value for the Service Desk attribute (value from ITSMExternalEnum)

fromVal - Object representation of the lowest value for the search range

toVal - Object representation of the highest value for the search range

Returns: AppCriterium object implementing the search condition defined

See Also: com.hp.ifc.util.marshall.AppCriterium

createRangeSearchCondition(int, String, String, Object, Object)

```
public com.hp.ifc.util.marshall.AppCriterium
    createRangeSearchCondition(int operator,
        java.lang.String entity, java.lang.String attrib,
        java.lang.Object fromVal, java.lang.Object toVal)
```

Creates a search condition (AppCriterium object) that specifies a range search.

This method to execute after setting the active import settings.

Parameters:

`operator` - logical operator for the combination of search conditions (value from `com.hp.ifc.util.marshall.AppWhereOperatorEnum`)
`entity` - String object for a mapped external entity name
`attrib` - String object for a mapped external attribute name
`fromVal` - Object representation of the lowest value for the search range
`toVal` - Object representation of the highest value for the search range

Returns: `AppCriterium` object implementing the search condition defined

See Also: `com.hp.ifc.util.marshall.AppCriterium`

createSearchCondition(int, int, AppOID[], Object, Object, boolean)

```
public com.hp.ifc.util.marshall.AppCriterium
    createSearchCondition(int operator, int
        critOperator, com.hp.ifc.types.AppOID[] attrib,
        java.lang.Object fromVal, java.lang.Object toVal,
        boolean negate)
```

Creates a tailor-made search condition (`AppCriterium` object). The search condition is created as specified by the parameters passed into this method. This allows for the creation of other search conditions than equality and range conditions.

Parameters:

`operator` - logical operator for the combination of search conditions (value from `com.hp.ifc.util.marshall.AppWhereOperatorEnum`)
`critOperator` - operator for the interpretation of search values (value from `com.hp.ifc.util.marshall.AppCriteriumOperatorEnum`).
`attrib` - `AppOID` array representing the Service Desk attribute. This array describes the actual attribute and its relation to the main entity

AppExternalAccess

`fromVal` - `Object` representation of the lowest value for the search range

`toVal` - `Object` representation of the highest value for the search range

`negate` - `true` when the search condition must be inverted (denied), otherwise `false`.

Returns: `AppCriterium` object implementing the search condition defined

See Also: `com.hp.ifc.util.marshal.AppCriterium`

createSearchCondition(int, int, long[], Object, Object, boolean)

```
public com.hp.ifc.util.marshal.AppCriterium
    createSearchCondition(int operator, int
        critOperator, long[] attrib, java.lang.Object
        fromVal, java.lang.Object toVal, boolean negate)
```

Creates a tailor-made search condition (`AppCriterium` object). The search condition is created as specified by the parameters passed into this method. This allows for the creation of other search conditions than equality and range conditions.

Parameters:

`operator` - logical operator for the combination of search conditions (value from `com.hp.ifc.util.marshal.AppWhereOperatorEnum`)

`critOperator` - operator for the interpretation of search values (value from

`com.hp.ifc.util.marshal.AppCriteriumOperatorEnum`).

`attrib` - long array for the Service Desk attribute. This array describes the actual attribute and its relation to the main entity (values from `ITSMEExternalEnum`)

`fromVal` - `Object` representation of the lowest value for the search range

`toVal` - `Object` representation of the highest value for the search range

negate - true when the search condition must be inverted (denied), otherwise false.

Returns: AppCriterium object implementing the search condition defined

See Also: com.hp.ifc.util.marshal.AppCriterium

createSearchCondition(int, int, String, String, Object, Object, boolean)

```
public com.hp.ifc.util.marshal.AppCriterium  
    createSearchCondition(int operator, int  
        critOperator, java.lang.String entity,  
        java.lang.String attrib, java.lang.Object fromVal,  
        java.lang.Object toVal, boolean negate)
```

Creates a tailor-made search condition (AppCriterium object). The search condition is created as specified by the parameters passed into this method. This allows for the creation of other search conditions than equality and range conditions.

This method to execute after setting the active import settings.

Parameters:

operator - logical operator for the combination of search conditions (value from

com.hp.ifc.util.marshal.AppWhereOperatorEnum)

critOperator - logical operator for the interpretation of search conditions (value from

com.hp.ifc.util.marshal.AppCriteriumOperatorEnum).

entity - String object for a mapped external entity name

attrib - String object for a mapped external attribute name

fromVal - Object representation of the lowest value for the search range

toVal - Object representation of the highest value for the search range

negate - true when the search condition must be inverted (denied), otherwise false.

AppExternalAccess

Returns: AppCriterium object implementing the search condition defined

See Also: `com.hp.ifc.util.marshall.AppCriterium`

disconnect()

```
public void disconnect()
```

Disconnects the client from the server.

find(AppOID, AppAttributeSelection, AppCriterium[])

```
public com.hp.ifc.types.AppOID[] find(com.hp.ifc.types.AppOID
    entOID,
    com.hp.ifc.util.marshall.AppAttributeSelection sel,
    com.hp.ifc.util.marshall.AppCriterium[] crits)
```

Returns an array of AppOID objects representing the instances of a given external entity that satisfy the given selection criteria. The attributes to retrieve are specified in the AppAttributeSelection object passed.

The individual instantiations can later be opened calling the open method.

This method to execute after setting the active import settings.

Parameters:

entOID - AppOID representation of an external entity

sel - AppAttributeSelection object specifying the attributes to retrieve

crits - array of AppCriterium objects describing conditions for selection

Returns: AppOID array for the selected external entity instantiations

See Also: `#createCriterium`,
`com.hp.ifc.util.marshall.AppCriterium`, `#realFind`,
[`setSettings\(AppOID\)`](#)

find(long, AppAttributeSelection, AppCriterium[])

```
public long[] find(long entity,
    com.hp.ifc.util.marshall.AppAttributeSelection sel,
    com.hp.ifc.util.marshall.AppCriterium[] crits)
```

Returns an array of long values representing the instances of a given external entity that satisfy the given selection criteria. The attributes to retrieve are specified in the `AppAttributeSelection` object passed.

The individual instantiations can later be opened calling the `open` method.

This method to execute after setting the active import settings.

Parameters:

`entity` - long value for an external entity

`sel` - `AppAttributeSelection` object specifying the attributes to retrieve

`crits` - array of `AppCriterium` objects describing conditions for selection

Returns: long array for the selected external entity instantiations

See Also: `#createCriterium`,
`com.hp.ifc.util.marshal.AppCriterium`,
`setSettings(AppOID)`

findMappedEntities(AppOID, AppCriterium[])

```
public com.hp.ifc.types.AppOID[]  
    findMappedEntities(com.hp.ifc.types.AppOID extOID,  
        com.hp.ifc.util.marshal.AppCriterium[] crits)
```

Returns an array of `AppOID` objects representing the instances of a given mapped external entity that satisfy the given selection criteria. The attributes to retrieve are determined from the mapped attributes defined on the external object.

The individual instantiations can later be opened calling the `openMappedEntity` method.

This method to execute after setting the active import settings.

Parameters:

`extOID` - `AppOID` representation of a mapped external entity name

`crits` - array of `AppCriterium` objects describing conditions for selection

Returns: `AppOID` array for the selected mapped external entity instantiations

AppExternalAccess

See Also: [AppMappedExternalEntity](#), [#createCriterium](#), [com.hp.ifc.util.marshall.AppCriterium](#), [#createAttributeSelection](#), [setSettings\(AppOID\)](#)

findMappedEntities(long, AppCriterium[])

```
public long[] findMappedEntities(long extEnt,
    com.hp.ifc.util.marshall.AppCriterium[] crits)
```

Returns an array of long values representing the instances of a given mapped external entity that satisfy the given selection criteria. The attributes to retrieve are determined from the mapped attributes defined on the external object.

The individual instantiations can later be opened calling the `openMappedEntity` method.

This method to execute after setting the active import settings.

Parameters:

`extEnt` - long value for a mapped external entity name

`crits` - array of `AppCriterium` objects describing conditions for selection

Returns: long array for the selected mapped external entity instantiations

See Also: [AppMappedExternalEntity](#), [#createCriterium](#), [com.hp.ifc.util.marshall.AppCriterium](#), [#createAttributeSelection](#), [setSettings\(AppOID\)](#)

findMappedEntities(String, AppCriterium[])

```
public com.hp.ifc.types.AppOID[]
    findMappedEntities(java.lang.String extName,
        com.hp.ifc.util.marshall.AppCriterium[] crits)
```

Returns an array of `AppOID` objects representing the instances of a given mapped external entity that satisfy the given selection criteria. The attributes to retrieve are determined from the mapped attributes defined on the external object.

The individual instantiations can later be opened calling the `openMappedEntity` method.

This method to execute after setting the active import settings.

Parameters:

extName - String object for a mapped external entity name

crits - array of AppCriterium objects describing conditions for selection

Returns: AppOID array for the selected mapped external entity instantiations

See Also: [AppMappedExternalEntity](#), [#createCriterium](#), [com.hp.ifc.util.marshall.AppCriterium](#), [setSettings\(AppOID\)](#)

findMappedEntitiesAsLong(String, AppCriterium[])

```
public long[] findMappedEntitiesAsLong(java.lang.String  
    extName, com.hp.ifc.util.marshall.AppCriterium[]  
    crits)
```

Returns an array of long values representing the instances of a given mapped external entity that satisfy the given selection criteria. The attributes to retrieve are determined from the mapped attributes defined on the external object.

The individual instantiations can later be opened calling the `openMappedEntity` method.

This method to execute after setting the active import settings.

Parameters:

extName - String object for a mapped external entity name

crits - array of AppCriterium objects describing conditions for selection

Returns: long array for the selected mapped external entity instantiations

See Also: [AppMappedExternalEntity](#), [#createCriterium](#), [com.hp.ifc.util.marshall.AppCriterium](#), [setSettings\(AppOID\)](#)

getAvailableLoadSettingNames()

```
public java.lang.String[] getAvailableLoadSettingNames()
```

AppExternalAccess

Returns an array of `Strings` for the names of the available load settings. One of these settings should be made the active setting.

This method to execute after succesfull login on the Service Desk system.

Returns: String array for the available load settings names

See Also: `#setSettings()`,
`com.hp.ifc.rep.ext.AppExternalLoadSetting`

getAvailableLoadSettings()

```
public com.hp.ifc.types.AppOID[] getAvailableLoadSettings()
```

Returns an array of `AppOID` objects representing the available load settings. One of these settings should be made the active setting.

This method to execute after succesfull login on the Service Desk system.

Returns: `AppOID` array for the available load settings

See Also: `setSettings(AppOID)`,
`com.hp.ifc.rep.ext.AppExternalLoadSetting`

getAvailableLoadSettingsAsLong()

```
public long[] getAvailableLoadSettingsAsLong()
```

Returns an array of `long` values representing the available load settings. One of these settings should be made the active setting.

This method to execute after succesfull login on the Service Desk system.

Returns: `long` array for the available load settings

See Also: `#setSettings()`,
`com.hp.ifc.rep.ext.AppExternalLoadSetting`

getExternalEntityInfo(AppOID)

```
public com.hp.ifc.rep.ext.AppExternalEntityInfo  
    getExternalEntityInfo(com.hp.ifc.types.AppOID ent)
```

Returns the `AppExternalEntityInfo` object specified by the `AppOID` object.

This method to execute after setting the active import settings.

Parameters:

ent - AppOID representation for the external entity

See Also: [setSettings\(AppOID\)](#),
com.hp.ifc.rep.ext.AppExternalEntityInfo

getExternalEntityInfo(long)

```
public com.hp.ifc.rep.ext.AppExternalEntityInfo  
    getExternalEntityInfo(long ent)
```

Returns the AppExternalEntityInfo object specified by the long value.

This method to execute after setting the active import settings.

Parameters:

ent - long representation for the external entity

See Also: [setSettings\(AppOID\)](#),
com.hp.ifc.rep.ext.AppExternalEntityInfo

getExternalEntityInfo(String)

```
public com.hp.ifc.rep.ext.AppExternalEntityInfo  
    getExternalEntityInfo(java.lang.String name)
```

Returns the AppExternalEntityInfo object specified by the name.

This method to execute after setting the active import settings.

Parameters:

name - String for the external entity name

See Also: [setSettings\(AppOID\)](#),
com.hp.ifc.rep.ext.AppExternalEntityInfo

getMappedAttributeAsLong(long)

```
public long[] getMappedAttributeAsLong(long entOID)
```

Returns an array of long values representing the attributes defined for a given external entity (AppExternalAttributeInfo objects). These are the mapped external attributes that are available for processing: they have been mapped to Service Desk attributes and defaults are defined for some of them.

This method to execute after setting the active import settings.

AppExternalAccess**Parameters:**

entOID - long representation of the AppExternalEntityInfo object for which the external attributes are to be returned

Returns: long array for the available external attributes

See Also: [setSettings\(AppOID\)](#),
com.hp.ifc.rep.ext.AppExternalAttributeInfo

getMappedAttributeNames(String)

```
public java.lang.String[]  
    getMappedAttributeNames(java.lang.String entity)
```

Returns an array of String objects for the attribute names defined for a given external entity (AppExternalAttributeInfo objects). These are the mapped external attributes that are available for processing: they have been mapped to Service Desk attributes and defaults are defined for some of them.

This method to execute after setting the active import settings.

Parameters:

entity - String object for the AppExternalEntityInfo object for which the external attributes are to be returned

Returns: String array for the available external entity names

See Also: [setSettings\(AppOID\)](#),
com.hp.ifc.rep.ext.AppExternalAttributeInfo

getMappedAttributes(AppOID)

```
public com.hp.ifc.types.AppOID[]  
    getMappedAttributes(com.hp.ifc.types.AppOID  
entOID)
```

Returns an array of AppOID objects representing the attributes defined for a given external entity (AppExternalAttributeInfo objects). These are the mapped external attributes that are available for processing: they have been mapped to Service Desk attributes and defaults are defined for some of them.

This method to execute after setting the active import settings.

Parameters:

entOID - AppOID representation of the
AppExternalEntityInfo object for which the external attributes
are to be returned

Returns: AppOID array for the available external attributes

See Also: `setSettings(AppOID)`,
`com.hp.ifc.rep.ext.AppExternalAttributeInfo`

getMappedEntities()

```
public com.hp.ifc.types.AppOID[] getMappedEntities()
```

Returns an array of AppOID objects representing the entities defined in the active import settings (AppExternalEntityInfo objects). These are the mapped external entities that are available for processing; they have been mapped to Service Desk entities and defaults are defined for them.

This method to execute after setting the active import settings.

Returns: AppOID array for the available external entities

See Also: `setSettings(AppOID)`,
`com.hp.ifc.rep.ext.AppExternalEntityInfo`

getMappedEntitiesAsLong()

```
public long[] getMappedEntitiesAsLong()
```

Returns an array of long values representing the entities defined in the active import settings (AppExternalEntityInfo objects). These are the mapped external entities that are available for processing; they have been mapped to Service Desk entities and defaults are defined for them.

This method to execute after setting the active import settings.

Returns: long array for the available external entities

See Also: `setSettings(AppOID)`,
`com.hp.ifc.rep.ext.AppExternalEntityInfo`

getMappedEntityNames()

```
public java.lang.String[] getMappedEntityNames()
```

Returns an array of Strings for the names of the entities defined in the active import settings (AppExternalEntityInfo objects). These are

AppExternalAccess

the mapped external entities that are available for processing: they have been mapped to Service Desk entities and defaults are defined for them.

This method to execute after setting the active import settings.

Returns: String array for the available external entity names

See Also: [setSettings\(AppOID\)](#),
`com.hp.ifc.rep.ext.AppExternalEntityInfo`

getMessage(Exception)

```
public void getMessage(java.lang.Exception message)
```

Re-throws an `Exception` object as an `ExternalException` object. This provides a standardized way of error handling throughout the Service Desk API environment.

Throws: [ExternalException](#) - when called

See Also: `com.hp.ifc.rep.AppMessage`,
`com.hp.ifc.rep.AppMsg`,
`com.hp.ifc.rep.AppDictionary`

listMappedEntities(AppOID, AppCriterion[])

```
public com.hp.ifc.types.AppOID[]
    listMappedEntities(com.hp.ifc.types.AppOID extOID,
        com.hp.ifc.util.marshall.AppCriterion[] crits)
```

Returns an array of `AppOID` objects representing the instances of a given mapped external entity that satisfy the given selection criteria.

This method to execute after setting the active import settings.

Parameters:

`extOID` - `AppOID` representation of a mapped external entity

`crits` - array of `AppCriterion` objects describing conditions for selection

Returns: `AppOID` array for the selected mapped external entity instantiations

See Also: [AppMappedExternalEntity](#), `#createCriterion`,
`com.hp.ifc.util.marshall.AppCriterion`,
`#realFind()`, [setSettings\(AppOID\)](#)

listMappedEntities(long, AppCriterium[])

```
public long[] listMappedEntities(long extOID,  
                                com.hp.ifc.util.marshal.AppCriterium[] crits)
```

Returns an array of long values representing the instances of a given mapped external entity that satisfy the given selection criteria.

This method to execute after setting the active import settings.

Parameters:

extOID - long value for a mapped external entity

crits - array of AppCriterium objects describing conditions for selection

Returns: long array for the selected mapped external entity instantiations

See Also: [AppMappedExternalEntity](#), [#createCriterium](#), [com.hp.ifc.util.marshal.AppCriterium](#), [setSettings\(AppOID\)](#)

listMappedEntities(String, AppCriterium[])

```
public com.hp.ifc.types.AppOID[]  
    listMappedEntities(java.lang.String extName,  
                       com.hp.ifc.util.marshal.AppCriterium[] crits)
```

Returns an array of AppOID objects representing the instances of a given mapped external entity that satisfy the given selection criteria.

This method to execute after setting the active import settings.

Parameters:

extName - String object for a mapped external entity name

crits - array of AppCriterium objects describing conditions for selection

Returns: AppOID array for the selected mapped external entity instantiations

See Also: [AppMappedExternalEntity](#), [#createCriterium](#), [com.hp.ifc.util.marshal.AppCriterium](#), [setSettings\(AppOID\)](#)

listMappedEntitiesAsLong(String, AppCriterium[])

AppExternalAccess

```
public long[] listMappedEntitiesAsLong(java.lang.String
    extName, com.hp.ifc.util.marshal.AppCriterium[]
    crits)
```

Returns an array of long values representing the instances of a given mapped external entity that satisfy the given selection criteria.

This method to execute after setting the active import settings.

Parameters:

extName - String object for a mapped external entity name

crits - array of AppCriterium objects describing conditions for selection

Returns: long array for the selected mapped external entity instantiations

See Also: [AppMappedExternalEntity](#), [#createCriterium](#), [com.hp.ifc.util.marshal.AppCriterium](#), [setSettings\(AppOID\)](#)

loggedIn()

```
public boolean loggedIn()
```

Returns true when connected into the Service Desk environment, otherwise false.

open(AppOID, AppAttributeSelection, AppOID)

```
public AppExternalEntity open(com.hp.ifc.types.AppOID entOID,
    com.hp.ifc.util.marshal.AppAttributeSelection sel,
    com.hp.ifc.types.AppOID instOID)
```

Creates an AppExternalEntity object for a given external entity and fills this with data on a given Service Desk entity instantiation. The attributes to expose are specified in the AppAttributeSelection object passed.

The new AppExternalEntity object can be used to change or remove the given entity instantiation in the Service Desk environment.

This method to execute after succesful login on the Service Desk system.

Parameters:

entOID - AppOID representation of an entity OID

`sel` - `AppAttributeSelection` object specifying the attributes to expose

`instOID` - `AppOID` object for a Service Desk entity instantiation

Returns: `AppExternalEntity` object for the given external entity instantiation

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: [AppExternalEntity](#)

open(long, AppAttributeSelection, long)

```
public AppExternalEntity open(long entity,  
                               com.hp.ifc.util.marshall.AppAttributeSelection sel,  
                               long instance)
```

Creates an `AppExternalEntity` object for a given external entity and fills this with data on a given Service Desk entity instantiation. The attributes to expose are specified in the `AppAttributeSelection` object passed.

The new `AppExternalEntity` object can be used to change or remove the given entity instantiation in the Service Desk environment.

This method to execute after succesful login on the Service Desk system.

Parameters:

`entOID` - long value for an entity OID.

`sel` - `AppAttributeSelection` object specifying the attributes to expose

`instOID` - long value for a Service Desk entity instantiation

Returns: `AppExternalEntity` object for the given external entity instantiation

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: [AppExternalEntity](#), [open\(AppOID, AppAttributeSelection, AppOID\)](#)

openMappedEntity(AppOID, AppOID)

AppExternalAccess

```
public AppMappedExternalEntity
    openMappedEntity(com.hp.ifc.types.AppOID extEnt,
        com.hp.ifc.types.AppOID instOID)
```

Creates an `AppMappedExternalEntity` object for a given mapped external entity and fills this with data on a given Service Desk entity instantiation. The attributes to expose are determined from the mapped attributes defined on the external object.

The new `AppMappedExternalEntity` object can be used to change or remove the given entity instantiation in the Service Desk environment.

This method to execute after the `setSettings` method.

Parameters:

`extEnt` - AppOID representation of a mapped external entity

`instOID` - AppOID object for a Service Desk entity instantiation

Returns: `AppMappedExternalEntity` object for the given external entity instantiation

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: `AppMappedExternalEntity`, `#setSettings()`, `#createAttribSelection()`

openMappedEntity(long, long)

```
public AppMappedExternalEntity openMappedEntity(long extEnt,
    long instOID)
```

Creates an `AppMappedExternalEntity` object for a given mapped external entity and fills this with data on a given Service Desk entity instantiation. The attributes to expose are determined from the mapped attributes defined on the external object.

The new `AppMappedExternalEntity` object can be used to change or remove the given entity instantiation in the Service Desk environment.

This method to execute after the `setSettings` method.

Parameters:

`extName` - long value for a mapped external entity

`instOID` - long value for a Service Desk entity instantiation

Returns: `AppMappedExternalEntity` object for the given external entity instantiation

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: `AppMappedExternalEntity`, `#setSettings()`, `#createAttribSelection()`

openMappedEntity(String, long)

```
public AppMappedExternalEntity  
    openMappedEntity(java.lang.String extName, long  
        instOID)
```

Creates an `AppMappedExternalEntity` object for a given mapped external entity and fills this with data on a given Service Desk entity instantiation. The attributes to expose are determined from the mapped attributes defined on the external object.

The new `AppMappedExternalEntity` object can be used to change or remove the given entity instantiation in the Service Desk environment.

This method to execute after the `setSettings` method.

Parameters:

`extEnt` - `String` object for a mapped external entity name

`instOID` - `long` value for a Service Desk entity instantiation

Returns: `AppMappedExternalEntity` object for the given external entity instantiation

Throws: `ExternalException` - thrown when Service Desk-specific error occurs

See Also: `AppMappedExternalEntity`, `#setSettings()`, `#createAttribSelection()`

reconnect()

```
public void reconnect()
```

Reconnects the client to the server.

setSettings(AppOID)

AppExternalAccess

```
public void setSettings(com.hp.ifc.types.AppOID set)
```

Sets the given set of import settings as the active set. This set of settings will be used until another active set is chosen.

This method to execute after succesful login on the Service Desk system.

Parameters:

set - AppOID representation of the data loading settings.

Throws: `ExternalException` - thrown when login not executed successfully.

See Also: `getAvailableLoadSettings()`,
`com.hp.ifc.rep.ext.AppExternalLoadSetting`

setSettings(long)

```
public void setSettings(long set)
```

Sets the given set of import settings as the active set. This set of settings will be used until another active set is chosen.

This method to execute after succesful login on the Service Desk system.

Parameters:

set - long value for the data loading settings

Throws: `ExternalException` - thrown when not logged in on the Service Desk environment

See Also: `getAvailableLoadSettingsAsLong()`,
`com.hp.ifc.rep.ext.AppExternalLoadSetting`

setSettings(String)

```
public void setSettings(java.lang.String set)
```

Sets the given set of import settings as the active set. This set of settings will be used until another active set is chosen.

This method to execute after succesful login on the Service Desk system.

Parameters:

set - String for the data loading settings name

Throws: `ExternalException` - thrown when not logged in on the Service Desk environment

See Also: [getAvailableLoadSettingNames\(\)](#),
`com.hp.ifc.rep.ext.AppExternalLoadSetting`

shutdown()

```
public static void shutdown()
```

Disconnects and shuts down the program.

Javadoc
AppExternalAccess

com.hp.ifc.ext AppExternalEntity

Syntax

```
public class AppExternalEntity extends java.lang.Object

java.lang.Object
|
+--com.hp.ifc.ext.AppExternalEntity
```

Direct Known Subclasses: [AppMappedExternalEntity](#)

Description

Access for external software to entities that are part of the HP OpenView Service Desk environment. The actual Service Desk entity is wrapped in this class in the form of an `IAppEntity` interface.

The use of this class relies on the availability of an instantiation of `AppExternalAccess`. This class provides some of the essential connections with Service Desk's Workflow Layer (searching, opening entities).

Instances of this class can save themselves independently of the `AppExternalAccess` class.

See Also: [AppExternalAccess](#), `com.hp.ifc.wf.IAppEntity`

Member Summary

Fields

[REVISION](#)

Constructors

[AppExternalEntity\(IAppEntity, AppExternalAccess\)](#)

Creates a new external entity wrapping a Service Desk object (implementation of `IAppEntity`).

Methods

Member Summary

<code>addSetReference(AppAttributeSelection, AppOID, AppOID)</code>	Adds a relation object to an EntitySetReference object related to this external entity.
<code>addSetReference(AppAttributeSelection, long, long, long)</code>	Adds a relation object to an EntitySetReference object related to this external entity.
<code>cancel()</code>	Rolls back the current transaction.
<code>delete()</code>	Deletes the wrapped Service Desk object and ends the current transaction.
<code>getEntityInfo()</code>	Returns the meta-information on the Service Desk entity wrapped (AppEntityInfo object).
<code>getKey()</code>	Returns the AppOID representation of the ObjectId attribute's value on the wrapped entity.
<code>getKeyAsLong()</code>	Returns the long value of the ObjectId attribute's value on the wrapped entity.
<code>getValue(AppOID)</code>	Returns a Object representation of the current value of the attribute passed.
<code>getValue(AppOID[])</code>	Returns a Object representation of the current value of the attribute passed.
<code>getValue(long)</code>	Returns a Object representation of the current value of the attribute passed.
<code>getValue(long[])</code>	Returns a Object representation of the current value of the attribute passed.
<code>save()</code>	Saves changes to the wrapped Service Desk object and ends the transaction.
<code>setValue(AppOID[], Object)</code>	Sets the the attribute to the indicated value.
<code>setValue(AppOID[], String)</code>	Sets the the attribute to the indicated value.
<code>setValue(AppOID, Object)</code>	Sets the the attribute to the indicated value.
<code>setValue(AppOID, String)</code>	Sets the the attribute to the indicated value.
<code>setValue(long[], Object)</code>	Sets the the attribute to the indicated value.
<code>setValue(long[], String)</code>	Sets the the attribute to the indicated value.
<code>setValue(long, Object)</code>	Sets the the attribute to the indicated value.
<code>setValue(long, String)</code>	Sets the the attribute to the indicated value.

Inherited Member Summary
<p>Methods inherited from class java.lang.Object</p> <p>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait</p>

Fields

REVISION

```
public static final java.lang.String REVISION
```

Constructors

AppExternalEntity(IAppEntity, AppExternalAccess)

```
public AppExternalEntity(com.hp.ifc.wf.IAppEntity entity,
    AppExternalAccess acc)
```

Creates a new external entity wrapping a Service Desk object (implementation of IAppEntity). The AppExternalEntity object created also contains an AppExternalAccess object.

This method will normally be called from an AppExternalAccess object.

Parameters:

- entity - IAppEntity implementation that is wrapped in this class
- access - AppExternalAccess object used to get access to the Service Desk system

See Also: [AppExternalAccess](#), [com.hp.ifc.wf.IAppEntity](#)

Methods

AppExternalEntity**addSetReference(AppAttributeSelection, AppOID, AppOID, AppOID)**

```
public void
    addSetReference(com.hp.ifc.util.marshal.AppAttributeSelection sel, com.hp.ifc.types.AppOID setOid,
        com.hp.ifc.types.AppOID refOid,
        com.hp.ifc.types.AppOID chlOid)
```

Adds a relation object to an `EntitySetReference` object related to this external entity. This change is persisted in the Service Desk database right away. This method is used for the addition of instances of many-to-many relations to the Service Desk system.

Parameters:

`sel` - `AppAttributeSelection` object, indicating the attributes of the relation that will be processed

`setOid` - `AppOID` object, indicating the current entity's attribute that references the relation to add

`refOid` - `AppOID` object, indicating the relation entity's attribute that references the child entity

`chlOid` - `AppOID` object, indicating the object for the child entity

Throws: `ExternalException` - thrown when an error occurs

See Also: `#isValidAttribute()`,
`com.hp.ifc.wf.AppEntitySet`,
`com.hp.ifc.wf.IAppEntity`

addSetReference(AppAttributeSelection, long, long, long)

```
public void
    addSetReference(com.hp.ifc.util.marshal.AppAttributeSelection sel, long setOid, long refOid, long
        chlOid)
```

Adds a relation object to an `EntitySetReference` object related to this external entity. This change is persisted in the Service Desk database right away. This method is used for the addition of instances of many-to-many relations to the Service Desk system.

Parameters:

`sel` - `AppAttributeSelection` object, indicating the attributes of the relation that will be processed

setOid - long value, indicating the current entity's attribute that references the relation to add

refOid - long value, indicating the relation entity's attribute that references the child entity

chOid - long value, indicating the object for the child entity

See Also: [addSetReference\(AppAttributeSelection, AppOID, AppOID, AppOID\)](#)

cancel()

```
public void cancel()
```

Rolls back the current transaction.

Throws: `ExternalException` - thrown when an error occurs

See Also: `IAppEntity.cancelEdit()`

delete()

```
public void delete()
```

Deletes the wrapped Service Desk object and ends the current transaction.

Throws: `ExternalException` - thrown when an error occurs

See Also: `IAppEntity.destroy()`

getEntityInfo()

```
public com.hp.ifc.rep.AppEntityInfo getEntityInfo()
```

Returns the meta-information on the Service Desk entity wrapped (`AppEntityInfo` object).

Returns: `AppEntityInfo` object for the Service Desk entity wrapped

Throws: `ExternalException` - thrown when an error occurs

See Also: `com.hp.ifc.rep.AppEntityInfo`

getKey()

```
public com.hp.ifc.types.AppOID getKey()
```

AppExternalEntity

Returns the AppOID representation of the ObjectId attribute's value on the wrapped entity.

Returns: AppOID representation for the Service Desk object Id

Throws: ExternalException - thrown when an error occurs

See Also: com.hp.ifc.types.AppOID

getKeyAsLong()

```
public long getKeyAsLong()
```

Returns the long value of the ObjectId attribute's value on the wrapped entity.

Returns: long value for the Service Desk object Id

Throws: ExternalException - thrown when an error occurs

getValue(AppOID)

```
public java.lang.Object getValue(com.hp.ifc.types.AppOID  
    attribute)
```

Returns a Object representation of the current value of the attribute passed. Before execution, a check is performed whether a valid attribute identifier has been passed.

Parameters:

attribute - AppOID representation of the entity's attribute

Returns: Object representation of the value of the attribute

Throws: ExternalException - thrown when an error occurs

See Also: IAppEntity.getValue(AppOID),
#isValidAttribute

getValue(AppOID[])

```
public java.lang.Object getValue(com.hp.ifc.types.AppOID[]  
    attributes)
```

Returns a Object representation of the current value of the attribute passed. The attribute is passed into this method as a nested attribute, i.e. an attribute of another entity that acts as an attribute of the main entity. These are either aggregations or references.

NOTE: attributes that are members of references to entitysets can not be retrieved using this method.

Parameters:

`attributes` - array of AppOID objects representing an entity's nested attribute

Returns: Object representation of the value of the nested attribute

Throws: ExternalException - thrown when an error occurs

See Also: [getValue\(AppOID\)](#), [#isValidAttribute](#)

getValue(long)

```
public java.lang.Object getValue(long attribute)
```

Returns a Object representation of the current value of the attribute passed. Before execution, a check is performed whether a valid attribute identifier has been passed.

Parameters:

`attribute` - long representation of the entity's attribute

Returns: Object representation of the value of the attribute

Throws: ExternalException - thrown when an error occurs

See Also: [getValue\(AppOID\)](#)

getValue(long[])

```
public java.lang.Object getValue(long[] attributes)
```

Returns a Object representation of the current value of the attribute passed. The attribute is passed into this method as a nested attribute, i.e. an attribute of another entity that acts as an attribute of the main entity. These are either aggregations or references.

NOTE: attributes that are members of references to entitysets can not be retrieved using this method.

Parameters:

`attributes` - array of long values representing an entity's nested attribute

Returns: Object representation of the value of the nested attribute

See Also: [getValue\(AppOID\[\]\)](#)

AppExternalEntity**save()**

```
public void save()
```

Saves changes to the wrapped Service Desk object and ends the transaction.

Throws: `ExternalException` - thrown when an error occurs

See Also: `IAppEntity.saveEdit()`

setValue(AppOID[], Object)

```
public void setValue(com.hp.ifc.types.AppOID[] attributes,  
                    java.lang.Object value)
```

Sets the the attribute to the indicated value. The attribute is passed into this method as a nested attribute, i.e. an attribute of another entity that acts as an entity of the main entity. These are either aggregations or references. With the first call on this method after creating or saving the current object, a new transaction is started.

NOTE: attributes that are members of references to entitysets can not be set using this method.

Parameters:

`attributes` - array of `AppOID` objects representing an entity's nested attribute

`value` - `Object` representation of the value

Throws: `ExternalException` - thrown when an error occurs

See Also: `#setValue(AppOID)`

setValue(AppOID[], String)

```
public void setValue(com.hp.ifc.types.AppOID[] attributes,  
                    java.lang.String value)
```

Sets the the attribute to the indicated value. Before the value is set, the `String` object is converted to the appropriate `Object` indicated by the Service Desk attribute definition. The attribute is passed into this method as a nested attribute, i.e. an attribute of another entity that acts as an entity of the main entity. These are either aggregations or references.

NOTE: attributes that are members of references to entitysets can not be set using this method.

Parameters:

attributes - array of AppOID objects representing an entity's nested attribute

value - String representation of the value

See Also: [setValue\(AppOID\[\], Object\)](#), #stringToServiceDeskObject

setValue(AppOID, Object)

```
public void setValue(com.hp.ifc.types.AppOID attribute,  
                    java.lang.Object value)
```

Sets the the attribute to the indicated value. Before execution, a check is performed whether a valid attribute identifier has been passed. With the first call on this method after creating or saving the current object, a new transaction is started.

Parameters:

attribute - AppOID representation of the entity's attribute

value - Object representation of the value

Throws: ExternalException - thrown when an error occurs

See Also: [IAppEntity.setValue\(AppOID, Object\)](#), #isValidAttribute, #beginEdit

setValue(AppOID, String)

```
public void setValue(com.hp.ifc.types.AppOID attribute,  
                    java.lang.String value)
```

Sets the the attribute to the indicated value. Before the value is set, the String object is converted to the appropriate Object indicated by the Service Desk attribute definition.

Parameters:

attribute - AppOID representation of the entity's attribute

value - String representation of the value

Throws: ExternalException - thrown when an error occurs

See Also: [setValue\(AppOID, Object\)](#), #stringToServiceDeskObject

AppExternalEntity**setValue(long[], Object)**

```
public void setValue(long[] attributes, java.lang.Object
                    value)
```

Sets the the attribute to the indicated value. The attribute is passed into this method as a nested attribute, i.e. an attribute of another entity that acts as an entity of the main entity. These are either aggregations or references. With the first call on this method after creating or saving the current object, a new transaction is started.

NOTE: attributes that are members of references to entitysets can not be set using this method.

Parameters:

`attributes` - array of long values representing an entity's nested attribute

`value` - Object representation of the value

See Also: [setValue\(AppOID\[\], Object\)](#)

setValue(long[], String)

```
public void setValue(long[] attributes, java.lang.String
                    value)
```

Sets the the attribute to the indicated value. Before the value is set, the `String` object is converted to the appropriate `Object` indicated by the Service Desk attribute definition. The attribute is passed into this method as a nested attribute, i.e. an attribute of another entity that acts as an entity of the main entity. These are either aggregations or references.

NOTE: attributes that are members of references to entitysets can not be set using this method.

Parameters:

`attributes` - array of long values representing an entity's nested attribute

`value` - String representation of the value

See Also: [setValue\(AppOID\[\], String\)](#)

setValue(long, Object)

```
public void setValue(long attribute, java.lang.Object value)
```

Sets the the attribute to the indicated value. Before execution, a check is performed whether a valid attribute identifier has been passed. With the first call on this method after creating or saving the current object, a new transaction is started.

Parameters:

attribute - long representation of the entity's attribute

value - Object representation of the value

Throws: `ExternalException` - thrown when an error occurs

See Also: `IAppEntity.setValue(AppOID, Object)`,
`#isValidAttribute`, `#beginEdit`

setValue(long, String)

```
public void setValue(long attribute, java.lang.String value)
```

Sets the the attribute to the indicated value. Before the value is set, the `String` object is converted to the appropriate `Object` indicated by the Service Desk attribute definition.

Parameters:

attribute - long representation of the entity's attribute

value - `String` representation of the value

Throws: `ExternalException` - thrown when an error occurs

See Also: `setValue(AppOID, Object)`

Javadoc
AppExternalEntity

com.hp.ifc.ext AppMappedExternalEntity

Syntax

```
public class AppMappedExternalEntity extends AppExternalEntity

java.lang.Object
|
+--AppExternalEntity
|
+--com.hp.ifc.ext.AppMappedExternalEntity
```

Description

Access for external software to entities that are part of the HP OpenView Service Desk environment. The access is regulated using the import mapping defined in the repository, so the entities are not approached as Service Desk entities, but as external entities. These external entities (can) have other names and Ids than the Service Desk entities.

The use of this class relies on the availability of an instantiation of `AppExternalAccess`. The latter class provides some of the essential connections with Service Desk's Workflow Layer (searching, opening entities, converting external entities to Service Desk entities).

See Also: [AppExternalAccess](#), [AppExternalEntity](#)

Member Summary

Fields

[REVISION](#)

Constructors

[AppMappedExternalEntity\(AppExternalEntityInfo, IAppEntity, AppExternalAccess\)](#)

Creates a new mapped external entity wrapping a Service Desk object (implementation of `IAppEntity`).

Methods

Javadoc
AppMappedExternalEntity

Member Summary

<code>addSetReference(String, String, AppAttributeInfo, AppOID, String[], String[])</code>	Adds a relation object to an EntitySetReference object related to this external entity.
<code>exists(String[], String[])</code>	Returns the AppOID object identifying a single Service Desk object that complies with ALL given search conditions.
<code>getValue(String)</code>	Gets the value for a given attribute.
<code>setExternalEntityInfo(AppExternalEntityInfo)</code>	Sets the external entity descriptive information.
<code>setValue(String, String)</code>	Sets the attribute passed to the String object.

Inherited Member Summary

Methods inherited from class AppExternalEntity

`addSetReference(AppAttributeSelection, AppOID, AppOID, AppOID)`, `addSetReference(AppAttributeSelection, long, long, long)`, `cancel()`, `delete()`, `getEntityInfo()`, `getKey()`, `getKeyAsLong()`, `getValue(AppOID)`, `getValue(AppOID[])`, `getValue(long)`, `getValue(long[])`, `save()`, `setValue(AppOID[], Object)`, `setValue(AppOID[], String)`, `setValue(AppOID, Object)`, `setValue(AppOID, String)`, `setValue(long[], Object)`, `setValue(long[], String)`, `setValue(long, Object)`, `setValue(long, String)`

Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`

Fields

REVISION

```
public static final java.lang.String REVISION
```


Constructors

AppMappedExternalEntity(AppExternalEntityInfo, IAppEntity, AppExternalAccess)

```
public
    AppMappedExternalEntity(com.hp.ifc.rep.ext.AppExternalEntityInfo entInfo, com.hp.ifc.wf.IAppEntity
        entity, AppExternalAccess access)
```

Creates a new mapped external entity wrapping a Service Desk object (implementation of `IAppEntity`). The `AppMappedExternalEntity` object created also contains an `AppExternalAccess` object. The `AppExternalEntityInfo` contained in it makes it self-describing.

This method will normally be called from an `AppExternalAccess` object.

Parameters:

`entInfo` - `AppExternalEntityInfo` object describing the external entity that is created.

`access` - `AppExternalAccess` object, used to get access to the Service Desk system.

Methods

addSetReference(String, String, AppAttributeInfo, AppOID, String[], String[])

```
public void addSetReference(java.lang.String relationName,
    java.lang.String childName,
    com.hp.ifc.rep.AppAttributeInfo oSet,
    com.hp.ifc.types.AppOID refOID,
    java.lang.String[] keys, java.lang.String[]
    values)
```

Adds a relation object to an `EntitySetReference` object related to this external entity. First, a check is executed whether the relation object already exists. This change is persisted in the Service Desk database right away.

AppMappedExternalEntity

This method is used for the addition of instances of many-to-many relations to the Service Desk system.

Parameters:

`relationName` - external attribute name denoting the `EntitySetRef` attribute involved. This name identifies the `AppExternalAttributeInfo` object for this relation

`childName` - external entity name denoting the related entity. This name identifies the `AppExternalEntityInfo` object for the related Service Desk object

`oSet` - `AppAttributeInfo` object denoting the Service Desk attribute that is the `EntitySetRef` relation

`refOid` - `AppOID` object, indicating the external entity's attribute that references the child entity.

`attrNames` - array of names of key attributes for the related entity. The names identify the `AppExternalAttributeInfo` objects for each external attribute that is a key attribute

`values` - array of `String` values for the key attributes

Throws: `ExternalException` - thrown when an error occurs

See Also: `com.hp.ifc.rep.ext.AppExternalEntityInfo`,
`com.hp.ifc.rep.ext.AppExternalAttributeInfo`,
`com.hp.ifc.rep.AppAttributeInfo`, `#exists()`,
`addSetReference(AppAttributeSelection, AppOID, AppOID, AppOID)`,
`AppExternalAccess#createEqualSearchCondition()`

exists(String[], String[])

```
public com.hp.ifc.types.AppOID exists(java.lang.String[]
    attrNames, java.lang.String[] values)
```

Returns the `AppOID` object identifying a single Service Desk object that complies with ALL given search conditions. Search conditions are defined by pairs of attribute names and values, passed into the method in two distinct `String` arrays.

The purpose of this method is to check for the existence of a given Service Desk object. This implies the creation of search conditions for all key

attributes defined with the entity's Import settings. No check is performed on the appropriateness or completeness of search criteria.

NOTE: It is the API programmers's responsibility to to assure that only key attributes are used in search conditions, and that all key attributes are used (suggested use of `AppExternalEntityInfo.getKeyNames()`).

Parameters:

`attrNames` - array of names of key attributes for the entity. The names identify the `AppExternalAttributeInfo` objects for each external attribute

`values` - array of `String` values for the key attributes

Returns: `AppOID` object for a single Service Desk object satisfying the search criteria

Throws: `ExternalComFailException` - when more than one instantiation is found that satisfies the search criteria, or another error occurs

See Also: `com.hp.ifc.rep.ext.AppExternalEntityInfo`, `AppExternalAccess#createEqualSearchCondition()`, `AppExternalAccess#listExternalEntities(String, AppCriterium[])`

getValue(String)

```
public java.lang.String getValue(java.lang.String attribute)
```

Gets the value for a given attribute. This value is returned as a formatted `String` using the formatting data stored within the Service Desk environment.

Parameters:

`attribute` - name for the external attribute for which the value is to be returned. This name identifies the `AppExternalAttributeInfo` object for the external attribute

Returns: `String` object representing the attribute value

Throws: `ExternalException` - thrown when an error occurs

See Also:

`com.hp.ifc.rep.ext.AppExternalAttributeInfo`, `AppExternalEntity.getValue`, `#getValidationClass`

Javadoc

AppMappedExternalEntity

setExternalEntityInfo(AppExternalEntityInfo)

```
public void  
    setExternalEntityInfo(com.hp.ifc.rep.ext.AppExternalEntityInfo entInfo)
```

Sets the external entity descriptive information.

Parameters:

entInfo - AppExternalEntityInfo object describing the mapped external entity

setValue(String, String)

```
public void setValue(java.lang.String attribute,  
    java.lang.String value)
```

Sets the attribute passed to the String object. The String object will be converted to a valid Service Desk object.

Parameters:

attribute - name for the external attribute for which the value is to be set. This name identifies the AppExternalAttributeInfo object for the external attribute

value - String object to set the attribute to

Throws: ExternalException - thrown when an error occurs

See Also:

[com.hp.ifc.rep.ext.AppExternalAttributeInfo](#),
[setValue\(long, String\)](#)

com.hp.ifc.ext AppExternalUtilities

Syntax

```
public class AppExternalUtilities extends java.lang.Object
```

```
java.lang.Object
|
+--com.hp.ifc.ext.AppExternalUtilities
```

Description

Some utility methods to use within the HP OpenView Service Desk external access context.

- converting a long array into an AppOID array
- converting an AppOID array into a long array
- generalized dealing with log files

Member Summary

Fields

REVISION

Constructors

AppExternalUtilities()

Methods

closeLog()

Flushes the contents of the log file writer to the log file, and closes both the log file and its writer.

flush()

Flushes the contents of the log file writer to the log file.

getDebug()

Returns the indicator for printing debug information.

logFileSet()

Returns true when a log file has been set, returns false otherwise.

setDebug(boolean)

Sets the indicator for writing debug information.

setLogFile(String)

Sets up the indicated file as the logfile for further use.

write(String)

Writes a given String to the log file's writer.

Member Summary

<code>writeLog(String)</code>	Writes a given <code>String</code> to the log file, preceded by two new lines (i.e.
<code>writeNewLine(String)</code>	Writes a given <code>String</code> to the log file's writer, preceded by a new line.

Inherited Member Summary

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`

Fields

REVISION

```
public static final java.lang.String REVISION
```

Constructors

AppExternalUtilities()

```
public AppExternalUtilities()
```

Methods

closeLog()

```
public static void closeLog()
```

Flushes the contents of the log file writer to the log file, and closes both the log file and its writer.

Throws: `IOException` - thrown when the operation can not be completed

See Also: [setLogFile\(String\)](#), `java.io.IOException`

flush()

```
public static void flush()
```

Flushes the contents of the log file writer to the log file.

Throws: `IOException` - thrown when the operation can not be completed

See Also: [setLogFile\(String\)](#), `java.io.IOException`

getDebug()

```
public static boolean getDebug()
```

Returns the indicator for printing debug information. This indicator is true when debug information must be printed, otherwise false.

Returns: true when debug information must be printed, otherwise false

logFileSet()

```
public static boolean logFileSet()
```

Returns true when a log file has been set, returns false otherwise.

Returns: true when a log file has been set, false otherwise

setDebug(boolean)

```
public static void setDebug(boolean debug)
```

Sets the indicator for writing debug information. This indicator causes additional information to be written to the log file when set.

Parameters:

debug - true when debug information must be printed, otherwise false

setLogFile(String)

AppExternalUtilities

```
public static void setLogFile(java.lang.String fileName)
```

Sets up the indicated file as the logfile for further use. The given file is checked for existence and writability and an `FileWriter` is initialized for it. This log file will be used for Data Exchange import logging all through the current JVM session.

When the file name `name` passed refers to an existing file, this file is deleted and re-created as an empty file. This only applies when the file is writable.

Parameters:

`fileName` - operating System name for the log file.

Throws: `IOException` - thrown when the preferred file can not be used or created.

See Also: `java.io.File`, `java.io.FileWriter`,
`java.io.IOException`

write(String)

```
public static void write(java.lang.String text)
```

Writes a given `String` to the log file's writer.

Parameters:

`text` - `String` to write to the log file

Throws: `IOException` - thrown when the operation can not be completed

See Also: [setLogFile\(String\)](#), `java.io.IOException`

writeLog(String)

```
public static void writeLog(java.lang.String text)
```

Writes a given `String` to the log file, preceded by two new lines (i.e. separated from any previous text by a blank line).

Parameters:

`text` - `String` to write to the log file

Throws: `IOException` - thrown when the operation can not be completed

See Also: [setLogFile\(String\)](#), `java.io.IOException`

writeNewLine(String)

```
public static void writeNewLine(java.lang.String text)
```

Writes a given `String` to the log file's writer, preceded by a new line.

Parameters:

`text` - `String` to write to the log file

Throws: `IOException` - thrown when the operation can not be completed

See Also: [setLogFile\(String\)](#), `java.io.IOException`

com.hp.ifc.ext ExternalException

Syntax

```
public class ExternalException extends java.lang.RuntimeException
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--java.lang.RuntimeException
            |
            +--com.hp.ifc.ext.ExternalException
```

All Implemented Interfaces: java.io.Serializable

Description

Exception class for use within the HP OpenView Service Desk external access context.

Member Summary

Fields

REVISION

Constructors

ExternalException(String)

Creates an external exception containing a customized message.

ExternalException(String, int)

Creates an external exception containing a customized message and a severity.

Methods

getSeverity()

Returns the severity, which defaults to 2 (critical).

Inherited Member Summary

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace`, `getLocalizedMessage`, `getMessage`, `printStackTrace`, `printStackTrace`, `printStackTrace`, `toString`

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`

Fields

REVISION

```
public static final java.lang.String REVISION
```

Constructors

ExternalException(String)

```
public ExternalException(java.lang.String message)
```

Creates an external exception containing a customized message.

Parameters:

`message` - `String` to be incorporated as the exception's message

ExternalException(String, int)

```
public ExternalException(java.lang.String exc, int severity)
```

Creates an external exception containing a customized message and a severity.

Parameters:

`message` - `String` to be incorporated as the exception's message

`severity` - `int` value indicating the message severity (value from `com.hp.ifc.rep.AppSeverityEnum`)

Methods

getSeverity()

```
public int getSeverity()
```

Returns the severity, which defaults to 2 (critical).

Returns: message severity (value from
`com.hp.ifc.rep.AppSeverityEnum`)

Javadoc
ExternalException

com.hp.ifc.ext AppSingleLoad

Syntax

```
public class AppSingleLoad extends java.lang.Object
```

```
java.lang.Object
|
+--com.hp.ifc.ext.AppSingleLoad
```

Description

Utility class for processing a single instantiation of an entity that is part of the HP OpenView Service Desk system. This is the class to use on `com.hp.ifc.rep.ext.AppExternalEntityInfo` objects, that is the external objects mapped to Service Desk entities by means of import mapping. This class:

- pre-processes the parameters passed on the command line, or calling the class's methods, including some checking;
- checks whether the instantiation exists;
- retrieves the instantiation's values form the Service Desk system;
- updates any values passed to it;
- persists the result into the Service Desk environment, either as an insert/update, or as a delete.

See Also: [AppExternalAccess](#), [AppMappedExternalEntity](#),
[com.hp.ifc.ext.imp.AppEntityHandler](#),
[com.hp.ifc.ev.AppHttpEvPost](#)

Member Summary

Fields

[REVISION](#)

Constructors

[AppSingleLoad\(\)](#) Creates a new AppSingleLoad object.

Javadoc
AppSingleLoad

Member Summary

<code>AppSingleLoad(AppExternalAccess)</code>	Creates a new AppSingleLoad object.
Methods	
<code>main(String[])</code>	Processes a single Service Desk entity when called from a command line.
<code>processEntity(String, String[], String[], boolean)</code>	Processes a single Service Desk entity, either inserting/updating or deleting the entity.
<code>processEntity(String, String, String[], String[], boolean)</code>	Processes a single Service Desk entity, either inserting/updating or deleting the entity.
<code>processEntity(String, String, String, String, String[], String[], boolean)</code>	Processes a single Service Desk entity, either inserting/updating or deleting the entity.
<code>setExternalAccess(AppExternalAccess)</code>	Sets the access attribute for this class.

Inherited Member Summary

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`

Fields

REVISION

```
public static final java.lang.String REVISION
```


Constructors

AppSingleLoad()

```
public AppSingleLoad()
```

Creates a new AppSingleLoad object.

AppSingleLoad(AppExternalAccess)

```
public AppSingleLoad(AppExternalAccess acc)
```

Creates a new AppSingleLoad object. The object created contains an AppExternalAccess object.

Parameters:

acc - AppExternalAccess object to be used

Methods

main(String[])

```
public static void main(java.lang.String[] argv)
```

Processes a single Service Desk entity when called from a command line. This method checks the parameters passed and then calls on one of the processEntity methods.

See Also: #processEntity()

processEntity(String, String[], String[], boolean)

```
public void processEntity(java.lang.String ent,  
                           java.lang.String[] attrs, java.lang.String[]  
                           values, boolean delete)
```

Processes a single Service Desk entity, either inserting/updating or deleting the entity. Processing is directed by the parameters passed to this method.

This version of processEntity() assumes a connection to the Service Desk environment has been established and a set of Import mappings has been selected.

AppSingleLoad**Parameters:**

`ent` - the name for the mapped external entity to process

`attrs` - array of names for the mapped entity's attributes to process. These are external attributes as defined using `com.hp.ifc.rep.ext.AppExternalAttribute`

`values` - array of `String` values for the attributes specified in `attrs`. Attributes and values are matched by their position in the arrays

`delete` - `true` when the entity specified is to be deleted, otherwise `false` (insert or update)

See Also: [AppMappedExternalEntity](#),
[AppExternalUtilities#writeLog\(\)](#)

processEntity(String, String, String[], String[], boolean)

```
public void processEntity(java.lang.String ent,  
                          java.lang.String setting, java.lang.String[]  
                          attrs, java.lang.String[] values, boolean delete)
```

Processes a single Service Desk entity, either inserting/updating or deleting the entity. Processing is directed by the parameters passed to this method.

This version of `processEntity()` assumes a connection to the Service Desk environment has been established.

Parameters:

`ent` - the name for the mapped external entity to process

`setting` - the name for the Import setting to use

`attrs` - array of names for the mapped entity's attributes to process. These are external attributes as defined using `com.hp.ifc.rep.ext.AppExternalAttribute`

`values` - array of `String` values for the attributes specified in `attrs`. Attributes and values are matched by their position in the arrays

`delete` - `true` when the entity specified is to be deleted, otherwise `false` (insert or update)

See Also: [com.hp.ifc.rep.ext.AppExternalLoadSetting](#),
[AppMappedExternalEntity](#),
[AppExternalUtilities#writeLog\(\)](#)

**processEntity(String, String, String, String, String, String[], String[],
boolean)**

```
public void processEntity(java.lang.String username,  
                          java.lang.String password, java.lang.String  
                          server, java.lang.String ent, java.lang.String  
                          setting, java.lang.String[] attrs,  
                          java.lang.String[] values, boolean delete)
```

Processes a single Service Desk entity, either inserting/updating or deleting the entity. Processing is directed by the parameters passed to this method.

This version of `processEntity()` creates a connection to the Service Desk environment and sets the Import mappings.

Parameters:

`accountName` - the logical account name as used in the Service Desk environment

`password` - the password used to authenticate the Service Desk account

`server` - the name or IP address of the application server

`ent` - the name for the mapped external entity to process

`setting` - the name for the Import setting to use

`attrs` - array of names for the mapped entity's attributes to process. These are external attributes as defined using `com.hp.ifc.rep.ext.AppExternalAttribute`

`values` - array of `String` values for the attributes specified in `attrs`. Attributes and values are matched by their position in the arrays

`delete` - true when the entity specified is to be deleted, otherwise false (insert or update)

See Also: [AppExternalAccess](#),
[com.hp.ifc.rep.ext.AppExternalLoadSetting](#),
[AppMappedExternalEntity](#),
[AppExternalUtilities#writeLog\(\)](#)

Javadoc

AppSingleLoad

setExternalAccess(AppExternalAccess)

```
public void setExternalAccess(AppExternalAccess acc)
```

Sets the access attribute for this class. The given `AppExternalAccess` object provides access to the Service Desk system.

Parameters:

`acc` - `AppExternalAccess` object to use

com.hp.ifc.ext AppRelationLoad

Syntax

```
public class AppRelationLoad extends java.lang.Object
```

```
java.lang.Object
|
+--com.hp.ifc.ext.AppRelationLoad
```

Description

Utility class used to process a single relation (association) to be added to the HP OpenView Service Desk system. This is the class to use on `com.hp.ifc.rep.ext.AppExternalEntityInfo` objects, that is external defined objects mapped to Service Desk entities by means of import mapping.

See Also: [AppExternalAccess](#), [AppMappedExternalEntity](#), `com.hp.ifc.rep.ext.AppExternalEntityInfo`

Member Summary

Fields

[REVISION](#)

Constructors

[AppRelationLoad\(AppExternalAccess\)](#) Creates a new AppSingleLoad object.

Methods

[processRelation\(String, String\[\], String, String\[\]\)](#) Adds a relation object to the Service Desk environment.

Inherited Member Summary

Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`

Fields

REVISION

```
public static final java.lang.String REVISION
```

Constructors

AppRelationLoad(AppExternalAccess)

```
public AppRelationLoad(AppExternalAccess acc)
```

Creates a new `AppSingleLoad` object. The object created contains an `AppExternalAccess` object.

Parameters:

`acc` - `AppExternalAccess` object to be used

Methods

processRelation(String, String[][], String, String, String[][])

```
public void processRelation(java.lang.String parentName,  
                             java.lang.String[][] parentKeys, java.lang.String  
                             relationName, java.lang.String childName,  
                             java.lang.String[][] childKeys)
```

Adds a relation object to the Service Desk environment. Before doing this, a check is performed to ascertain whether the relation already exists.

Both `EntityRefs` and `EntitySetRefs` are processed by this method.

Parameters:

`parentName` - the name for the Service Desk object where the relation originates

`parentKeys` - two-dimensional array of key names and values for the parent object

`relationName` - the name for the attribute that defines the relation

`childName` - the name for the the Service Desk object where the relation points to

`childKeys` - two-dimensional array of key names and values for the child object

See Also: [AppMappedExternalEntity](#),
`com.hp.ifc.rep.ext.AppExternalEntityInfo`,
`com.hp.ifc.rep.ext.AppExternalAttributeInfo`,
`com.hp.ifc.rep.AppAttributeInfo`

Javadoc
AppRelationLoad

Glossary

A

API Application programming interface. An interface that enables programmatic access to an application.

attribute A characteristic or property associated with a system, network, or other item. OpenView items represent system, network, and personnel resources by modeling and providing information on the attributes (properties and state) of an object. An attribute has a name and a value. An attribute is a place holder in which a specific value is held to provide information about the state of the item. For example, incident description, or employee name.

B

business layer The business layer contains rules that determine how an operation is validated or completed. It communicates with the data access layer and the workflow layer. The business layer works on the database side to find, load, delete, or save data and then present only the data asked for to the workflow layer.

business rules A set of hooks that overrule the standard behavior defined for an entity in the repository and inherited from the super class. The hooks are collected in a class that extends AppEntity.

C

configuration item An item belonging to the technical infrastructure of an organization. A configuration item may consist of other configuration items, and may be part of other configuration items. For example, a PC, an application program, a network, a work space with desks and chairs. Configuration items are stored in the application database.

class An encapsulated collection of data and methods to operate on the data. A class may be instantiated to produce an object that is an instance of the class. The class defines how the objects should be accessed by other objects, whether the object is public, and under what circumstances it can be created.

D

data access layer The data access layer provides access to the data in the database. It communicates between the application's object model and the relational representation of what exists in the underlying database.

data mapping model Describes how entities and their attributes in the object model are mapped to the database. The application server reads it when started. It is used by the business layer to persist the data. The data mapping model is persisted in the `ifc_tables` and the `ifc_columns`.

E

encapsulate An object-oriented programming technique that makes an object's data private or protected (that is: hidden) and allows programmers to access and manipulate that data only through method calls. Done well, encapsulation reduces bugs and promotes reusability and modularity of classes. This technique is also known as data hiding.

entity An entity is a logical collection of attributes. It is the basic item in the object model of Service Desk. In the object model, the entity is the top item.

I

IDL Interface definition language. Used to describe CORBA object interfaces. It describes the interfaces that client objects call and object implementations provide.

IFC ITSM foundation classes. These classes together form the base of the Service Desk application, called the kernel. Service Desk-specific features have not been incorporated in the IFC, making it possible to develop other database-based applications with it.

instance An object that exposes a particular interface. An object is an instance of an interface if it provides the operations, signatures, and semantics specified by that interface. An object is an instance of an implementation if its behavior is provided by that implementation.

interface A specification, written in IDL, of the operations and attributes that an object provides.

K

kernel See IFC

N

non-UI account Service Desk accounts granting users access to the Service Desk application but not the user interface. Non-UI accounts can be created for users who will not be using the Service Desk interfaces but need access for other purposes. The number of named user accounts is usually limited by the licensing agreement, while non-UI accounts are not.

O

object An encapsulated software unit consisting of both state (data) and behavior (code). Objects have attributes, methods, and events. Attributes make up the data that describes an object. Methods are the functions an object can perform. Events are the functions an object can perform in response to another event. In some object

models an object is an instance of a class as specified in some object-modeling languages.

object model A model of the application's entities, attributes, and their relations.

P

presentation layer The presentation layer contains the user interfaces. Information from the presentation layer is passed to the workflow layer.

R

repository The repository is a collection of data that governs the behavior of the application. The repository contains information about: the object model, data mapping model, user-interface configuration, role information, accounts, pools, labels, and messages.

U

user interface configuration

Includes the data form definitions, data view definitions, and user-specific definitions.

W

workflow layer The workflow layer consists of classes on the client. It operates directly under the presentation layer and is responsible for handling the data flow between the business layer and the presentation layer (or the API). If data is changed by the presentation layer or the API the workflow layer check that the data is logically correct.

wrapper A type of glueware that is used to attach other software components together. A wrapper may encapsulate a single system, often a data source, to make it usable in some new way that the unwrapped system was not. Wrappers can be used to expose all or some of the functionality of the thing they are wrapping, and present a simplified or standard interface to make a component more available.

Index

A

- addSetReference(AppAttributeSelection, AppOID, AppOID, AppOID) -
 - com.hp.ifc.ext.AppExternalEntity.addSetReference(com.hp.ifc.util.marshall.AppAttributeSelection, com.hp.ifc.types.AppOID, com.hp.ifc.types.AppOID, com.hp.ifc.types.AppOID), 102
 - addSetReference(AppAttributeSelection, long, long, long) -
 - com.hp.ifc.ext.AppExternalEntity.addSetReference(com.hp.ifc.util.marshall.AppAttributeSelection, long, long, long), 102
 - addSetReference(String, String, AppAttributeInfo, AppOID, String[], String[]) -
 - com.hp.ifc.ext.AppMappedExternalEntity.addSetReference(java.lang.String, java.lang.String, com.hp.ifc.rep.AppAttributeInfo, com.hp.ifc.types.AppO-1921897313, 113
 - AppExternalAccess, 22, 29
 - AppExternalAccess -
 - com.hp.ifc.ext.AppExternalAccess, 65
 - AppExternalAccess(AppSession) -
 - com.hp.ifc.ext.AppExternalAccess.AppExternalAccess(com.hp.ifc.wf.AppSession), 69
 - AppExternalAccess(String, String) -
 - com.hp.ifc.ext.AppExternalAccess.AppExternalAccess(java.lang.String, java.lang.String), 70
 - AppExternalEntity, 23, 24
 - AppExternalEntity -
 - com.hp.ifc.ext.AppExternalEntity, 99
 - AppExternalEntity(IAppEntity, AppExternalAccess) -
 - com.hp.ifc.ext.AppExternalEntity.AppExternalEntity(com.hp.ifc.wf.IAppEntity, com.hp.ifc.ext.AppExternalAccess), 101
 - AppExternalUtilities -
 - com.hp.ifc.ext.AppExternalUtilities, 117
 - AppExternalUtilities() -
 - com.hp.ifc.ext.AppExternalUtilities.AppExternalUtilities(), 118
 - application server, connecting to, 29
 - AppMappedExternalEntity, 23, 24
 - AppMappedExternalEntity -
 - com.hp.ifc.ext.AppMappedExternalEntity, 111
 - AppMappedExternalEntity(AppExternalEntityInfo, IAppEntity, AppExternalAccess) -
 - com.hp.ifc.ext.AppMappedExternalEntity.AppMappedExternalEntity(com.hp.ifc.rep.ext.AppExternalEntityInfo, com.hp.ifc.wf.IAppEntity, com.hp.ifc.ext.AppExternalAccess), 113
 - AppRelationLoad, 25
 - AppRelationLoad -
 - com.hp.ifc.ext.AppRelationLoad, 133
 - AppRelationLoad(AppExternalAccess) -
 - com.hp.ifc.ext.AppRelationLoad.AppRelationLoad(com.hp.ifc.ext.AppExternalAccess), 134
 - AppSingleLoad, 25
 - AppSingleLoad -
 - com.hp.ifc.ext.AppSingleLoad, 127
 - AppSingleLoad() -
 - com.hp.ifc.ext.AppSingleLoad.AppSingleLoad(), 129
 - AppSingleLoad(AppExternalAccess) -
 - com.hp.ifc.ext.AppSingleLoad.AppSingleLoad(com.hp.ifc.ext.AppExternalAccess), 129
 - architecture, runtime, 14
 - attribute values, getting, 24, 45
 - attribute values, setting, 24, 47
- ## B
- business layer, 14
- ## C
- cancel() -
 - com.hp.ifc.ext.AppExternalEntity.cancel(), 103
 - closeLog() -
 - com.hp.ifc.ext.AppExternalUtilities.closeLog(), 118
 - com.hp.ifc.ext - com.hp.ifc.ext, 63
 - ComFailException, 25
 - connecting to application server, 22, 29
 - create(AppOID, AppAttributeSelection) -
 - com.hp.ifc.ext.AppExternalAccess.create(com.hp.ifc.type
-

- es.AppOID,
 - com.hp.ifc.util.marshall.AppAttributeSelection), 70
 - create(long, AppAttributeSelection) - com.hp.ifc.ext.AppExternalAccess.create(long, com.hp.ifc.util.marshall.AppAttributeSelection), 71
 - createEqualSearchCondition(int, AppOID, Object) - com.hp.ifc.ext.AppExternalAccess.createEqualSearchCondition(int, com.hp.ifc.types.AppOID, java.lang.Object), 72
 - createEqualSearchCondition(int, AppOID[], Object) - com.hp.ifc.ext.AppExternalAccess.createEqualSearchCondition(int, com.hp.ifc.types.AppOID[], java.lang.Object), 72
 - createEqualSearchCondition(int, long, Object) - com.hp.ifc.ext.AppExternalAccess.createEqualSearchCondition(int, long, java.lang.Object), 73
 - createEqualSearchCondition(int, long[], Object) - com.hp.ifc.ext.AppExternalAccess.createEqualSearchCondition(int, long[], java.lang.Object), 73
 - createEqualSearchCondition(int, String, String, Object) - com.hp.ifc.ext.AppExternalAccess.createEqualSearchCondition(int, java.lang.String, java.lang.String, java.lang.Object), 74
 - createMappedEntity(AppOID) - com.hp.ifc.ext.AppExternalAccess.createMappedEntity(com.hp.ifc.types.AppOID), 74
 - createMappedEntity(long) - com.hp.ifc.ext.AppExternalAccess.createMappedEntity(long), 75
 - createMappedEntity(String) - com.hp.ifc.ext.AppExternalAccess.createMappedEntity(java.lang.String), 75
 - createRangeSearchCondition(int, AppOID, Object, Object) - com.hp.ifc.ext.AppExternalAccess.createRangeSearchCondition(int, com.hp.ifc.types.AppOID, java.lang.Object, java.lang.Object), 76
 - createRangeSearchCondition(int, AppOID[], Object, Object) - com.hp.ifc.ext.AppExternalAccess.createRangeSearchCondition(int, com.hp.ifc.types.AppOID[], java.lang.Object, java.lang.Object), 76
 - createRangeSearchCondition(int, long, Object, Object) - com.hp.ifc.ext.AppExternalAccess.createRangeSearchCondition(int, long, java.lang.Object, java.lang.Object), 78
 - createRangeSearchCondition(int, long[], Object, Object) - com.hp.ifc.ext.AppExternalAccess.createRangeSearchCondition(int, long[], java.lang.Object, java.lang.Object), 77
 - createRangeSearchCondition(int, String, String, Object, Object) - com.hp.ifc.ext.AppExternalAccess.createRangeSearchCondition(int, java.lang.String, java.lang.String, java.lang.Object, java.lang.Object), 78
 - createSearchCondition(int, int, AppOID[], Object, Object, boolean) - com.hp.ifc.ext.AppExternalAccess.createSearchCondition(int, int, com.hp.ifc.types.AppOID[], java.lang.Object, java.lang.Object, boolean), 79
 - createSearchCondition(int, int, long[], Object, Object, boolean) - com.hp.ifc.ext.AppExternalAccess.createSearchCondition(int, int, long[], java.lang.Object, java.lang.Object, boolean), 80
 - createSearchCondition(int, int, String, String, Object, Object, boolean) - com.hp.ifc.ext.AppExternalAccess.createSearchCondition(int, int, java.lang.String, java.lang.String, java.lang.Object, java.lang.Object, boolean), 81
- D**
- data access layer, 14

Index

delete() -
com.hp.ifc.ext.AppExternal
Entity.delete(), 103

E

entities, finding, 36
entities, identifying, 31
error presentation, 25, 58
exists(String[], String[]) -
com.hp.ifc.ext.AppMappedE
xternalEntity.exists(java.lan
g.String[],
java.lang.String[]), 114
ExternalException, 25
ExternalException -
com.hp.ifc.ext.ExternalExce
ption, 123
ExternalException(String) -
com.hp.ifc.ext.ExternalExce
ption.ExternalException(jav
a.lang.String), 124
ExternalException(String, int) -
com.hp.ifc.ext.ExternalExce
ption.ExternalException(jav
a.lang.String, int), 124

F

find(AppOID,
AppAttributeSelection,
AppCriterion[]) -
com.hp.ifc.ext.AppExternal
Access.find(com.hp.ifc.types.
AppOID,
com.hp.ifc.util.marshal.App
AttributeSelection,
com.hp.ifc.util.marshal.App
Criterion[]), 82
find(long,
AppAttributeSelection,
AppCriterion[]) -
com.hp.ifc.ext.AppExternal
Access.find(long,

com.hp.ifc.util.marshal.App
AttributeSelection,
com.hp.ifc.util.marshal.App
Criterion[]), 82
finding entities, 36
finding entity instantiations, 36
finding instantiations, 23
findMappedEntities(AppOID,
AppCriterion[]) -
com.hp.ifc.ext.AppExternal
Access.findMappedEntities(
com.hp.ifc.types.AppOID,
com.hp.ifc.util.marshal.App
Criterion[]), 83
findMappedEntities(long,
AppCriterion[]) -
com.hp.ifc.ext.AppExternal
Access.findMappedEntities(l
ong,
com.hp.ifc.util.marshal.App
Criterion[]), 84
findMappedEntities(String,
AppCriterion[]) -
com.hp.ifc.ext.AppExternal
Access.findMappedEntities(j
ava.lang.String,
com.hp.ifc.util.marshal.App
Criterion[]), 84
findMappedEntitiesAsLong(Str
ing, AppCriterion[]) -
com.hp.ifc.ext.AppExternal
Access.findMappedEntities
AsLong(java.lang.String,
com.hp.ifc.util.marshal.App
Criterion[]), 85
flush() -
com.hp.ifc.ext.AppExternal
Utilities.flush(), 119

G

getAvailableLoadSettingNames(
) -
com.hp.ifc.ext.AppExternal
Access.getAvailableLoadSet
tingNames(), 85
getAvailableLoadSettings() -
com.hp.ifc.ext.AppExternal
Access.getAvailableLoadSet
tings(), 86
getAvailableLoadSettingsAsLon
g() -
com.hp.ifc.ext.AppExternal
Access.getAvailableLoadSet
tingsAsLong(), 86
getDebug() -
com.hp.ifc.ext.AppExternal
Utilities.getDebug(), 119
getEntityInfo() -
com.hp.ifc.ext.AppExternal
Entity.getEntityInfo(), 103
getExternalEntityInfo(AppOID)
-
com.hp.ifc.ext.AppExternal
Access.getExternalEntityInf
o(com.hp.ifc.types.AppOID),
86
getExternalEntityInfo(long) -
com.hp.ifc.ext.AppExternal
Access.getExternalEntityInf
o(long), 87
getExternalEntityInfo(String) -
com.hp.ifc.ext.AppExternal
Access.getExternalEntityInf
o(java.lang.String), 87
getKey() -
com.hp.ifc.ext.AppExternal
Entity.getKey(), 103
getKeyAsLong() -
com.hp.ifc.ext.AppExternal
Entity.getKeyAsLong(), 104

Index

- getMappedAttributeAsLong(long) -
 - com.hp.ifc.ext.AppExternalAccess.getMappedAttributeAsLong(long), 87
 - getMappedAttributeNames(String) -
 - com.hp.ifc.ext.AppExternalAccess.getMappedAttributeNames(java.lang.String), 88
 - getMappedAttributes(AppOID) -
 - com.hp.ifc.ext.AppExternalAccess.getMappedAttributes(com.hp.ifc.types.AppOID), 88
 - getMappedEntities() -
 - com.hp.ifc.ext.AppExternalAccess.getMappedEntities(), 89
 - getMappedEntitiesAsLong() -
 - com.hp.ifc.ext.AppExternalAccess.getMappedEntitiesAsLong(), 89
 - getMappedEntityNames() -
 - com.hp.ifc.ext.AppExternalAccess.getMappedEntityNames(), 89
 - getMessage(Exception) -
 - com.hp.ifc.ext.AppExternalAccess.getMessage(java.lang.Exception), 90
 - getSeverity() -
 - com.hp.ifc.ext.ExternalException.getSeverity(), 125
 - getting attribute values, 24, 45
 - getting instantiations, 23, 42
 - getValue(AppOID) -
 - com.hp.ifc.ext.AppExternalEntity.getValue(com.hp.ifc.types.AppOID), 104
 - getValue(long) -
 - com.hp.ifc.ext.AppExternalEntity.getValue(long), 105
 - getValue(long[]) -
 - com.hp.ifc.ext.AppExternalEntity.getValue(long[]), 105
 - getValue(String) -
 - com.hp.ifc.ext.AppMappedExternalEntity.getValue(java.lang.String), 115
- H**
- handling errors, 25, 58
- I**
- identifying entities, attributes, 22, 31
 - IFC, 20
 - instantiation, processing, 25
 - instantiations, finding, 23
 - instantiations, getting, 23, 42
 - instantiations, processing, 54
 - instantiations, saving, 24, 51
 - ITSM Foundation Classes, 20
 - ITSMExternalEnumeration, 23
- J**
- javadoc, 61
- K**
- kernel, 20
- L**
- listMappedEntities(AppOID, AppCriterium[]) -
 - com.hp.ifc.ext.AppExternalAccess.listMappedEntities(om.hp.ifc.types.AppOID, com.hp.ifc.util.marshal.AppCriterium[]), 90
 - listMappedEntities(long, AppCriterium[]) -
 - com.hp.ifc.ext.AppExternalAccess.listMappedEntities(long, com.hp.ifc.util.marshal.AppCriterium[]), 91
 - listMappedEntities(String, AppCriterium[]) -
 - com.hp.ifc.ext.AppExternalAccess.listMappedEntities(java.lang.String, com.hp.ifc.util.marshal.AppCriterium[]), 91
 - listMappedEntitiesAsLong(String, AppCriterium[]) -
 - com.hp.ifc.ext.AppExternalAccess.listMappedEntitiesAsLong(java.lang.String, com.hp.ifc.util.marshal.AppCriterium[]), 91
 - logFileSet() -
 - com.hp.ifc.ext.AppExternalUtilities.logFileSet(), 119
 - loggedIn() -
 - com.hp.ifc.ext.AppExternalAccess.loggedIn(), 92
- M**
- main(String[]) -
 - com.hp.ifc.ext.AppSingleLoaded.main(java.lang.String[]), 129
- O**
- open(AppOID, AppAttributeSelection, AppOID) -
 - com.hp.ifc.ext.AppExternal
-

Index

- Access.open(com.hp.ifc.types.AppOID, com.hp.ifc.util.marshal.AppAttributeSelection, com.hp.ifc.types.AppOID), 92
- open(long, AppAttributeSelection, long) - com.hp.ifc.ext.AppExternalAccess.open(long, com.hp.ifc.util.marshal.AppAttributeSelection, long), 93
- openMappedEntity(AppOID, AppOID) - com.hp.ifc.ext.AppExternalAccess.openMappedEntity(com.hp.ifc.types.AppOID, com.hp.ifc.types.AppOID), 93
- openMappedEntity(long, long) - com.hp.ifc.ext.AppExternalAccess.openMappedEntity(long, long), 94
- openMappedEntity(String, long) - com.hp.ifc.ext.AppExternalAccess.openMappedEntity(java.lang.String, long), 95
- P**
- presentation layer, 14
- processEntity(String, String, String, String, String[]) - com.hp.ifc.ext.AppSingleLoad.processEntity(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String[], boolean), 131
- processEntity(String, String, String[], String[], boolean) - com.hp.ifc.ext.AppSingleLoad.processEntity(java.lang.String, java.lang.String, java.lang.String[], boolean), 130
- processEntity(String, String[], String[], boolean) - com.hp.ifc.ext.AppSingleLoad.processEntity(java.lang.String, java.lang.String[], boolean), 129
- processing instantiations, 25
- processRelation(String, String[], String, String, String[]) - com.hp.ifc.ext.AppRelationLoad.processRelation(java.lang.String, java.lang.String[], java.lang.String, java.lang.String, java.lang.String[]), 134
- R**
- reconnect(), 95
- requirements, 16
- REVISION - com.hp.ifc.ext.AppExternalAccess.REVISION, 69
- REVISION - com.hp.ifc.ext.AppExternalEntity.REVISION, 101
- REVISION - com.hp.ifc.ext.AppExternalUtilities.REVISION, 118
- REVISION - com.hp.ifc.ext.AppMappedExternalEntity.REVISION, 112
- REVISION - com.hp.ifc.ext.AppRelationLoad.REVISION, 134
- REVISION - com.hp.ifc.ext.AppSingleLoad.REVISION, 128
- REVISION - com.hp.ifc.ext.ExternalException.REVISION, 124
- S**
- save() - com.hp.ifc.ext.AppExternalEntity.save(), 106
- saving instantiations, 24, 51
- Service Desk architecture, 14
- setDebug(boolean) - com.hp.ifc.ext.AppExternalUtilities.setDebug(boolean), 119
- setExternalAccess(AppExternalAccess) - com.hp.ifc.ext.AppSingleLoad.setExternalAccess(com.hp.ifc.ext.AppExternalAccess), 132
- setExternalEntityInfo(AppExternalEntityInfo) - com.hp.ifc.ext.AppMappedExternalEntity.setExternalEntityInfo(com.hp.ifc.rep.ext.AppExternalEntityInfo), 116
- setLogFile(String) - com.hp.ifc.ext.AppExternalUtilities.setLogFile(java.lang.String), 119
-

Index

- setSettings(AppOID) -
 - com.hp.ifc.ext.AppExternalAccess.setSettings(com.hp.ifc.types.AppOID), 95
 - setSettings(long) -
 - com.hp.ifc.ext.AppExternalAccess.setSettings(long), 96
 - setSettings(String) -
 - com.hp.ifc.ext.AppExternalAccess.setSettings(java.lang.String), 96
 - setting attribute values, 24, 47
 - setValue(AppOID, Object) -
 - com.hp.ifc.ext.AppExternalEntity.setValue(com.hp.ifc.types.AppOID, java.lang.Object), 107
 - setValue(AppOID, String) -
 - com.hp.ifc.ext.AppExternalEntity.setValue(com.hp.ifc.types.AppOID, java.lang.String), 107
 - setValue(AppOID[], Object) -
 - com.hp.ifc.ext.AppExternalEntity.setValue(com.hp.ifc.types.AppOID[], java.lang.Object), 106
 - setValue(AppOID[], String) -
 - com.hp.ifc.ext.AppExternalEntity.setValue(com.hp.ifc.types.AppOID[], java.lang.String), 106
 - setValue(long, Object) -
 - com.hp.ifc.ext.AppExternalEntity.setValue(long, java.lang.Object), 108
 - setValue(long, String) -
 - com.hp.ifc.ext.AppExternalEntity.setValue(long, java.lang.String), 109
 - setValue(long[], Object) -
 - com.hp.ifc.ext.AppExternalEntity.setValue(long[], java.lang.Object), 108
 - setValue(long[], String) -
 - com.hp.ifc.ext.AppExternalEntity.setValue(long[], java.lang.String), 108
 - setValue(String, String) -
 - com.hp.ifc.ext.AppMappedExternalEntity.setValue(java.lang.String, java.lang.String), 116
 - single instantiation processing, 25, 54
- W**
- workflow layer, 14
 - write(String) -
 - com.hp.ifc.ext.AppExternalUtilities.write(java.lang.String), 120
 - writeLog(String) -
 - com.hp.ifc.ext.AppExternalUtilities.writeLog(java.lang.String), 120
 - writeNewLine(String) -
 - com.hp.ifc.ext.AppExternalUtilities.writeNewLine(java.lang.String), 121