SONARIS/Framework

Java Client Programming Guide

Version 1.4.3

ORIMOS S.A.

Innere Gueterstrasse 4
6304 Zug (Switzerland)

Phone	+41-41-725-3570
Fax	+41-41-725-3579
Web	www.ORIMOS.com
Email	info@ORIMOS.com

ORIMOS Group
Berlin · Frankfurt · London · Zurich

Contents

Preface

Product Conventions

SONARIS/Framework is case sensitive. For example, if an application is being run with the name *MyApp* then client programs must connect to it using *MyApp* as the application name; *myapp*, *MYAPP* or variations are not acceptable. Application database data source names may or may not be case sensitive, and this depends on the database system and operating system being used. Case sensitivity applies to anything that has a name.

Typographic Conventions

The following typesetting conventions apply throughout this guide:

- When referring to programs `this font` is used. Programs and libraries are referred to without any operating system specific file extension. For example, the program `safcontrol.exe` as supplied on a Windows system is referred to simply as `safcontrol`.
- When referring to files or directories `this font` is used. Note that the directory path separator is always **/**.
- Examples of the use of, and output from, command line programs are shown in `this font`. In cases where long lines have to be split over a number of document lines to keep them readable the \ character denotes continuation on a following line. For example:

  ```
  >safbase -m FROG MyApp \
  SAFSQLSERVER;UID=sa;DATABASE=saf_myapp safw32odbc
  ```

- When showing samples of code `this font` is used.
- When referring to items on dialogs, menu items or other named entities this font is used. Sub-menus and menu items are separated by the ▷ character (e.g. Programs ▷ SONARIS)

1 Introduction

This guide will show you how to create a simple **SONARIS/Framework** java client program using the java interface. The client program gets values from the server and writes them and any updates to the screen as they are received.

The **SONARIS/Framework** application service that the client communicates with is called **crossrate**. This application calculates exchange rates for the British Pound (GBP), Euro (EUR), Swiss Franc (CHF) and Canadian Dollar (CAD) against one of these currencies. The currency that the cross rates are currently being calculated against is referred to as the *base* currency.

1.1 Prerequisites

Before continuing, make sure you have read the *SONARIS/Framework - Overview and Concepts*. In order to view and compile the example code presented in this text, please install the following subsets from the **SONARIS/Framework** kit:

- **Runtime Environment**
- **Server Environment**
- **Java Client Interface**
- **Software Development Kit**

1.2 The SONARIS/Framework Programming Environment

There are a number of techniques and concepts which are common to all areas of **SONARIS/Framework** programming whether you are creating **SONARIS/Framework** clients or your own custom SAOs.

1.2.1 Interfaces

The majority of the services made available by the **SONARIS/Framework** APIs are accessed through *interfaces*. In C++, an interface is a class which just contains pure virtual methods. Java has interfaces as part of the language. COM also has the concept of interfaces built in.

The interfaces used in the **SONARIS/Framework** APIs fall into two groups:

- *active* interfaces where **SONARIS/Framework** provides the implementation of the interface and methods to create and destroy instances of that implementation.
- *listener* interfaces for which the user must create an implementation on which **SONARIS/Framework** can then make calls to inform the user of asynchronous events. When such an event occurs **SONARIS/Framework** calls the appropriate method in the user's implementation of the interface; these methods are said to be “called back”.

Object Management

In the C++ APIs, all the active interfaces have a create() method which the user should call to get a new instance of an object which implements the interface, and a corresponding destroy() method which should be called when the user no longer requires the instance.

SONARIS/Framework provides a means to automate the creation and deletion of these objects within a block of code.The SAF::AutoDestroy template creates an instance of the specified interface and then destroys it when it goes out of scope. This has the additional benefit of making the deletion of the object exception proof.

Backwards Compatibility

Interfaces are at the heart of **SONARIS/Framework**s backwards compatibility strategy as they allow the underlying `safsystem` DLL to be upgraded without the need to re-compile or re-link any programs that use the **SONARIS/Framework** APIs.

Active interfaces support backwards compatibility because **SONARIS/Framework** provides their implementation. This means that the implementation of the interface can be changed without affecting any already compiled code. The interface can also be safely extended as long as the new methods are added at the end of the interface definition.

Providing backwards compatibility for listener interfaces is more complicated. If a listener interface has a method added (it must be at the end) then **SONARIS/Framework** must guarantee not the call the method if the user's implementation of the interface does not contain that method. **SONARIS/Framework** is able to make this decision on the basis of the API version numbers automatically provided to it when the user initialises the API. **SONARIS/Framework** knows what "shape" the listener interfaces were for any previous version of the API and so knows whether any particular user's implementation of an interface contains the relevant method.

1.2.2 Data Types and Status

Individual items of data used by **SONARIS/Framework** are held in an implementation of the ISAFValue active interface. An ISAFValue can hold any of the data types supported by **SONARIS/Framework** as both single values and arrays.

An ISAFValue also has a status associated with the data it holds (if any) which indicates the quality of the data or the reason why it does not have any data. Only if the status is SAF_VALID or SAF_STALE does the ISAFValue have data available.

1.2.3 Compilation and Linking

Refer to Appendix B for details of compiling code for use with **SONARIS/Framework**.

2 A Worked Example

2.1 S/AF Java API Concepts

The **SONARIS/Framework** client API is asynchronous because the events that the client needs to know about can happen at any time. A client is told about events when they happen and therefore does not need to poll to detect changes in the application.

SONARIS/Framework uses listener interfaces to tell a client about events. When an event occurs **SONARIS/Framework** calls the appropriate method in an implementation of the interface. Two interfaces are used for communication of events.

- ISAFClientListener is used for notification of events related to the whole client API. This may be a change in the status of the server application, or a change in the connection to the server.
- ISAOListener is used for notification of events that relate to a specific SAO or a command sent to the server.

In these examples, we have chosen to implement both interfaces by deriving from adapter classes ISAFClientAdapter and SAOAdapter and only implementing the methods required for the samples. In an attempt to make the samples as easy to understand as possible much of the clutter associated with error handling has been removed.

2.1.1 Thread Model

The **SONARIS/Framework** java client API supports two threading models. The model used is determined by the choice of initilisation method called on the client instance. An application can only use one type of thread model. An attempt to create further clients of a different model will be rejected.

Client Thread Event Delivery

Events are delivered on a thread created by the **SONARIS/Framework** client API. This would normally be the model used by a non GUI application that wished to receive events. It *can* be used by a Swing application, but the application would then have to make use of synchronized methods and javax.swing.SwingUtilities.invokeLater() to ensure that information was passed to UI components on the event thread. Clients of this type are created by calling SAFClient.createClient() and initialising with ISAFClient.initialise().

Swing Thread Event Delivery

Events are delivered on the Swing event thread. In this mode there is no concern about Swing threading issues. *However*, modal dialog boxes will cause the Swing event thread to block, and thus halt the delivery of information to the client. It is recommended that such dialogs be created and run in their own thread to avoid this problem. They will retain their modality, but allow events to be received on the main event thread. Clients of this type are created by calling SAFClient.createSwingClient() and initialising with ISAFClient.initialiseSwingClient().

2.2 XRate Description

This document describes two clients using this **SONARIS/Framework** application:

- JClientSample writes **crossrate** data to the console.
- JSwingClientSample is a Swing application and displays **crossrate** data in a javax.swing.JTable.

2.2.1 Establishing a Connection

To start communicating with a **SONARIS/Framework** application service, you create a **SONARIS/Framework** client object. In these examples this is done in the class main() method. There is no reason why the clients may not be created and connected elsewhere, with the proviso that Swing clients should not be created and initialised on the main Swing event thread. You must specify a ISAFClientListener. The ISAOListener is optional. However, it *must* be specified in a request or command if a global listener has not been provided. In these examples we have chosen to specify the SAO listener when the client is created.

Once the client has been created it can be initialised. This call makes the connection to the **SONARIS/Framework** server supplying the application. Initialisation can fail for a variety of reasons including the fact that the application is not running or the login details have been rejected by the application. Some client programs may attempt to retry initialisation again with different credentials.

2.2.2 Requesting SAOs

To request **crossrate** SAOs we specify their paths in the hierarchy in much the same way as paths are specified for accessing files in a file system. **SONARIS/Framework** provides the ISAOSpec interface and com.orimos.saf.client.SAOSpec class to assist in constructing SAO paths.

In the samples the ISAOSpecs are built level by level. There are methods on ISAOSpec that take complete paths but by building a path level by level side steps the need for separators.

The SAOs that we need fall into two groups:

- The Bid and Ask price (expressed in US dollars) for each of the currencies. These SAOs are found under `CrossRates/TestData/Forex`.
- The calculated Mid price (expressed in US dollars) of each currency which is used to calculate the cross rate against the base currency. These SAOs are found under `Cross-Rates/Calculations`.

When an ISAOSpec is complete the SAO it represents is requested by calling ISAFClient.request(). Each request and command sent to a **SONARIS/Framework** server must have an Id associated with it. The Id returned as part of the response to a command and can be used to identify specific requests and commands where the same listener is used for multiple requests and commands. The request id is a java long. ISAFClient.getRid() provides a method for obtaining a unique Id to use in a request. The Id starts at ISAFClient.NO_REQUEST + 1 and is incremented after each call to getRid(). The upper limit on requests is therefore java.lang.Long.MAX_VALUE or, 2^{63} -1 requests through each ISAFClient instance. There is no requirement to get Ids in this manner. A client can use any long values. If an Id is reused while still in use, the original request or command will be cancelled prior to the new request or command being issued.

The samples also make an association between the request Id and other information they require to be able to perform their tasks efficiently. The console application tracks its requests by creating a JClientSample.FieldRequest to hold request information, and the Swing sample uses JSwingClientSample.FieldRequest to hold the row and column in the model where the data for the field should be stored.

2.3 The Console Client

The console client requests its information and writes the values to the screen as they are received. It makes no use of moust of the callbacks that are available to it. The following sections describe its use of the callbacks.

2.3.1 Client Event Notification

- JClientSample.MyClientListener.statusChange() Invoked when there is a change in the state of the connection to the application The client prints out any message which is more serious than informational.

- JClientSample.MyClientListener..noLicence() is invoked if there is no licence available or it is invalid. If this happens then there is little the client can do except exit gracefully.

- JClientSample.MyClientListener.batchStart() and JClientSample.MyClientListener.batchEnd() are invoked at the beginning and end of a group of SAO events. The client prints a message. This shows that some clients might be able to optimise their operation by performing some functions during the batch methods and not on each SAO event.

2.3.2 SAO Event Notification

- JClientSample.MySAOListener.response() calls the update() method as there is no need for response specific processing.

- JClientSample.MySAOListener.update() The method may be invoked because of a change in the SAO's value or status. It is important to check the status of an SAO before trying to do anything with it. The examines the SAOs state and prints its value. See /ref xrate_print_sao

- JClientSample.MySAOListener.failed() is invoked when a requested SAO can not be found or a *command* fails. As this application does not use any commands then if this method is called it must be because one of the requested SAOs does not exist. Note that this method can safely delete the FieldRequest instance because **SONARIS/Framework** will not call any further event notification functions for this request.

The following methods are not overridden because we are not interested in receiving notification of changes in hierarchy structure.

- ISAOListener.childAdded() is invoked when a new child SAO is added to an SAO that has been requested.

- ISAOListener.childRemoved() is invoked when a child SAO is removed from an SAO that has been requested.

- ISAOListener.childRenamed() is invoked when the name of a child SAO is changed.

- ISAOListener.cancelled() because we do not cancel requests and do not need to be notified that a cancellation has been processed.

- ISAOListener.succeeded() is not overridden because it is invoked to inform us that a *command* has completed successfully, and we do not issue any commands.

2.3.3 Interpreting SAO Contents

JClientSample.MySAOListener.update() looks at the supplied com.orimos.saf.sao.ISAO and prints a textual representation of its contents. The first check it makes is whether the SAO contains any data; com.orimos.saf.sao.ISAO.isAvailable() returns true if the SAO contains a value.

If the SAO has a value then it may be *Valid* or *Stale* (i.e. it is not up-to-date because, for example, the **SONARIS/Framework** application has lost its connection to its external source of information); com.orimos.saf.sao.ISAO.isStale() returns true if the SAO contains a stale value. Next we switch on the type of data that the SAO contains and use the appropriate accessor method to get the value from the SAO. For double we want to format with a limited number of decimal places

Finally, we print out the name of the currency and field, its value and whether it is stale or not.

2.4 The Swing Client

The Swing client uses a javax.swing.JTable to display the xrate information. Almost all of the work is performed in JSwingClientSample.MyTableModel, an javax.swing.table.AbstractTableModel derived class.

2.4.1 Client Event Notification Methods

Most of the client level events are left unimplemented to provide as unclutered interface as possible.

- JSwingClientSample.MyClientListener.statusChange() is invoked when the status of the **SONARIS/Framework** application changes and displays a message box for a status change more serious than informational.

- JSwingClientSample.MyClientListener.noLicence() is invoked if there is no licence available or it is invalid. If this happens then there is little the application can do except exit gracefully after displaying a message box.

2.4.2 SAO Event Notification

The following methods are implemented to handle response, update and failure event notifications.

- JSwingClientSample.MyTableModel.MySAOListener.response() is invoked when an SAO is successfully obtained. In this application we treat initial responses the same as updates so we just invoke JSwingClientSample.MySAOListener.update().
- JSwingClientSample.MyTableModel.MySAOListener.update() is invoked when an update for an SAO is received. An update may be a change in the SAO's value or a change in its status. In this implementation, the status of the SAO is checked and the display modified to reflect the changed status or value.
- JSwingClientSample.MyTableModel.MySAOListener.failed() is invoked when a requested SAO can not be found or a *command* SAO fails. This application does not use any command SAOs so this method is called because one of the requested SAOs does not exist.

The following methods have not been implemented because we have no need to handle them as the application makes no calls that would result in events being raised.

- ISAOListener.childAdded()
- ISAOListener.childRemoved()
- ISAOListener.childRenamed()
- ISAOListener.succeeded()
- ISAOListener.cancelled()

2.5 Client Status Changes

We discussed the notification function that gets invoked when the status of the **SONARIS/Framework** Client changes in S/AF Java API Concepts but other things happen when the Client's status changes. **SONARIS/Framework** automatically fails over requests when a **SONARIS/Framework** application stops and restarts.

Try stopping and restarting the **crossrate** application while one of the sample applications is running.

2.5.1 Client Disconnect

When a client disconnects from the server (e.g. if the server is stopped or the network connection is lost) then eventually the ISAFClientListener.statusChange() implementation will be invoked. But before this happens, the ISAOListener.update() implementation will have been called for each SAO that the application has on request. The contents of these updates will indicate that there is no longer data available for their respective SAOs. This means that the application does not need to do anything explicit to modify the state of the data it is holding - as long as its implementation of ISAOListener.update() reacts correctly to the SAO's value not being available then status change processing comes virtually for free.

2.5.2 Client Reconnect

When a client reconnects to the server (after being disconnected) then the ISAFClient-Listener.statusChange() implementation will be invoked. However, it is not necessary to manually re-request the SAOs when this happens. Instead, the ISAOListener.update() implementation is automatically invoked for each of the existing SAO requests to provide the newly received value from the server. Once again, the application does not need to do anything explicit to handle the re-connection event.

Appendix A Class Documentation

A.1 com.orimos.saf.sao.IResult Interface Reference

A.1.1 Detailed Description

An interface used to group pieces of information supplied via callbacks on the ISAOListener and ISAOPropertyListener interfaces in response to request method calls on ISAFClient.

Definition at line 62 of file IResult.java.

Public Member Functions

- ISAFValue getCommandResult ()

Package Functions

- long getId ()
- ISAO getSAO ()
- Long **getLongId** ()
- ISAFClient getClient ()
- ISAOPropertyListener getPropertyListener ()
- ISAOListener getListener ()
- ISRecord getRecord ()

A.1.2 Member Function Documentation

ISAFValue com.orimos.saf.sao.IResult.getCommandResult ()

Returns:
A command result supplied as an ISAFValue if the request involved the execution of some command.

long com.orimos.saf.sao.IResult.getId () `[package]`

Returns:
The requestId appropriate to the result being delivered.

ISAO com.orimos.saf.sao.IResult.getSAO () `[package]`

Returns:
An ISAO if an SAO was requested.

ISAFClient com.orimos.saf.sao.IResult.getClient () `[package]`

Returns:
The ISAFClient on which the request was made.

ISAOPropertyListener com.orimos.saf.sao.IResult.getPropertyListener () `[package]`

Returns:
The ISAOPropertyListener to which the IResult is being delivered.

ISAOListener com.orimos.saf.sao.IResult.getListener () `[package]`

Returns:
The ISAOListener to which the IResult is being delivered.

ISRecord com.orimos.saf.sao.IResult.getRecord () `[package]`

Returns:
An ISRecord associated with a record request. null if it wasn't a record request.

The documentation for this interface was generated from the following file:

- IResult.java

A.2 com.orimos.saf.client.ISAFClient Interface Reference

A.2.1 Detailed Description

Each client instantiates a ISAFClient object when it wants to communicate with a named SAF application.

<p/> You need to understand the thread model. All callbacks occur on a callback thread; for a Swing client this is the main Swing event dispatch thread. For non Swing clients a thread is created specifically for callbacks. Whenever a request is made (this term includes cancels and terminations) it is transferred to the callback thread before it is actioned. If a request is made from the callback thread then it can either be actioned immediately or a message can be sent for action later. The client interface gives you the choice of how it is to behave in this respect. The default is "pureAsync" mode; in this mode requests will always be deferred. <p/> Non-pure-Async mode is more efficient but there are more restrictions on the things which can be done inside callbacks. You should be particularly careful not to cancel requests from with a callback relating to a DIFFERENT request - if your application needs to do this then you must use pure-Async mode. <p/> Obviously there are things that are not wise (particularly when done from within a callback) regardless of the model; cancelling a request id twice, re-using a request id between the two cancels, terminating an ISAFClient and then placing requests or cancels on it. <p/> For Swing clients time consuming operations should not be performed directly during during callbacks. Particularly if they require a response from a user interface. Eventually, the server dise of S/AF will decide that the client has 'gone away' and disconnect from it. This is particlarly applicable to modal dialogs that could be left waiting for a response for significant periods of time. Where modal dialogs need to be displayed from a callback, they should be started in a separate thread if at all possible. <p/> A request call will normally succeed and callbacks will occur later. Failure of the request itself will usually indicate a programming error, some inconsistency in parameters or the ISAOSpec does not contain enough information. In non-pureAsync mode a callback may occur before the return from the request if you make the request from the callback thread - this is usually the case on a cancel and may occur on a request when the information requested is already available. Indeed a snapshot request might have both the response and the cancelled callback processed in this way. You should not re-use a request-Id until after you have seen the cancelled callback of the first use but (currently, and this may change) doing so will result in the original request being cancelled (you will get a cancelled callback) and the new request replacing it. <p/> A client can request the same SAO multiple times. Each multiple request will be treated individually and each must be cancelled using the ID that was supplied when the request was made. <p/> Most work is expected to be done on the callback thread. The responses to all commands will occur on the callback thread. You should not access an ISAO outside the callback thread unless you have locked the client first. All ISAFClient objects share a single callback thread. <p/> SAOs are always requested via an ISAOSpec. For convenience ISAFClient provides an instance of an ISAOSpec implementation so, provided you do not make reentrant requests in this way, you may use this instance.

Definition at line 198 of file ISAFClient.java.

Public Member Functions

- void cancelTable (long requestId)

  Cancel a table request.

Static Public Attributes

- final int OK = 0

  Return codes for calls to initialise and initialiseSwingClient.

- final int **RETRY** = 1
- final int **NO_SONARIS_MESSAGING** = 2
- final int **NO_SAF_APPLICATION** = 3
- final int **NO_SAF_LOGIN** = 4
- final long NO_REQUEST = Long.MIN_VALUE

  A request Id which can never be obtained from this client unless more than Long.MAX_VALUE requests are made.

- final String PROPERTY_NAME_PATH = ”com.orimos.saf.”

  Convenient for building what follows...

- final String SAFVARS_PROPERTY = PROPERTY_NAME_PATH + ”saf.safvars”

  The properties key for the safvars properties object.

- final String SAFVARS_PROPERTIES_FILE_NAME = ”safvars.properties”

  The name of the properties file that has its values loaded into the safvars properties object.

- final String SAFVARS_LOCATION_PROPERTY = PROPERTY_NAME_PATH + ”safvars”

  The property path of the system properties property that can be used to define a new location for the SAF properties file.

- final String BIN_DIR_PROPERTY = PROPERTY_NAME_PATH + ”binDir”

  The bin directory where programs and configuration are expected to be loaded from.

- final String SAFDIR_PROPERTY = PROPERTY_NAME_PATH + ”safdir”

  The directory into which the current SAF installation has been installed.

- final int STOP = 1

  Flag to stop an application.

- final int START = 0

  Flag to start an application.

- final int EXIT = 2

  Flag to exit an application.

- final int INCLUDE_INPUTS = 0x1

  These constants control the content of exported XML or an imported hierarchy.

- final int **INCLUDE_VALUES** = 0x2
- final int **INCLUDE_TRANSIENTS** = 0x4
- final int **INCLUDE_EXTERNAL_INPUTS** = 0x8
- final int **INCLUDE_PERMISSIONS** = 0x20
- final int NORMAL_EXPORT

  These are the settings of the control parameters for most exports.

- final int NORMAL_IMPORT

  This defines an import with the most useful control parameters set.

- final int SEND_AND_WAIT = 0
- final int SEND_TO_RUNNING = 1
- final int CANCEL_UNLESS_COMPLETE = 2
- final int IMPORT_CREATE_TRANSIENT = 0x10

  Flags that control how the import is performed.

- final int **IMPORT_IGNORE_MISMATCHED_LIBS** = 0x40
- final int **IMPORT_IGNORE_MISMATCHED_INPUTS** = 0x80
- final int **IMPORT_CREATE_PERSISTENT** = 0x100
- final int **LAST_CHILD** = 0
- final int **FIRST_CHILD** = 1
- final int **NEXT_SIBLING** = 2
- final int **PREV_SIBLING** = 3
- final int **USE_NEXT** = 5

Package Functions

- int initialise ()

  Initialise this instance.

- int initialiseSwingClient ()

  Initialise this instance.

- void setTimeout (int secs)

  Sets a timeout for initialise.

- long getRid ()

  Provide a request Id.

- boolean hasLicense ()
- int terminate ()

  Terminate the client.

- void lock ()

  Locks in preparation for accessing ISAOs.

- void unlock ()

  Frees a lock.

- ISAO find (ISAOSpec id)

  Finds an SAO locally.

- ISAO find (String name)

  Finds an SAO.

- ISAO find (ISAO base, String name)

  Finds an SAO by name relative from a given SAO.

- void cancel (long requestId)

  Cancel a request for information from an SAO.

- boolean request (ISAOSpec id, long requestId, boolean data, boolean includeChildren, boolean snapshot, ISAOListener saoListener, ISAOPropertyListener propListener)

Request an SAO.

- boolean activate (ISAO sao, long requestId, boolean data, boolean includeChildren, boolean snapshot, ISAOListener saoListener, ISAOPropertyListener propListener)

  Activate an SAO.

- boolean modify (long requestId, ISAOPropertyListener propListener)

  Change the ISAOPropertyListener associated with a request.

- boolean setSAOValue (long requestId, ISAOSpec id, ISAFValue value, ISAOListener saoListener)

  Use this method to change the value of an SAO.

- boolean setSAOValue (long requestId, ISAOSpec id, ISAFValue value, ISAOListener saoListener, boolean complete)

  Use this method to change the value of an SAO.

- boolean copyTree (long requestId, ISAOSpec source, String name, ISAOSpec parent, ISAOSpec next, boolean all, boolean transientCopy, ISAOListener saoListener)

  Use this method to copy a tree of SAOs.

- boolean moveTree (long requestId, ISAOSpec source, ISAOSpec parent, ISAOSpec next, ISAOListener saoListener)

  Use this method to move a tree of SAOs.

- boolean modifyTree (long requestId, ISAOSpec source, ISAOSpec destination, ISAOListener saoListener, int control)

  Use this method to modify a tree of SAOs to match a template.

- boolean invokeCommandSAO (ISAOSpec id, long requestId, ISAFProperties params, ISAOListener saoListener)

  This method invokes the remoteCommand method of an SAO.

- boolean addSAO (long requestId, SAOFType type, String name, ISAOSpec parent, ISAOSpec next, boolean persistent, ISAOListener saoListener)

  Create and insert an SAO by functional type.

- boolean addSAO (long requestId, String libName, String saoName, String name, ISAOSpec parent, ISAOSpec next, boolean persistent, ISAOListener saoListener)

Create and insert an SAO by name.

- boolean removeSAO (long requestId, ISAOSpec sao, ISAOListener saoListener)

  Use this method to remove an SAO.

- boolean linkSAO (long requestId, ISAOSpec destination, ISAOSpec source, int index, boolean overwrite, ISAOListener saoListener)

  Use this method to create a dependancy between two SAOs.

- boolean unlinkSAO (long requestId, ISAOSpec destination, ISAOSpec source, int index, ISAOListener saoListener)

  Use this method to remove a dependancy between two SAOs.

- ISAFPermissioning getPermissioning ()

  Provides access to the permissioning information.

- boolean changePassword (long requestId, String doer, String oldpassword, String user, String newpassword, ISAOListener saoListener)

  Change a user's password.

- void encryptPassword (String user, String pw, ISAFValue output)

  Encrypts a username and password into an ISAFValue; special function to be used when changing a user's password via a command SAO.

- boolean newCredentials (String user, String pw)

  Replace the username and password supplied to SAFClient#createClient SAFClient#createSwingClient or previous calls to this method with those specified.

- boolean verifyCredentials (String user, String pw)

  Determine if a user name and password are valid and whether or not the user has system privilege.

- void cancelSAOId (SAOId saoId)

  Cancel request for information from an SAO by SAOId.

- boolean reserveSAO (long requestId, ISAOSpec dest, ISAOListener saoListener)

  Use this method to mark an SAO at the server.

- boolean releaseSAO (long requestId, ISAOSpec dest, ISAOListener saoListener)

Use this method to release a reserved SAO at the server.

- void addClientListener (ISAFClientListener listener)

  Add a new client listener to the current set.

- void removeClientListener (ISAFClientListener listener)

  Remove a client listener from this client.

- int getContainerState (ISAOSpec spec)

  Determine the availability of a container.

- boolean isContainerAvailable (short cid)

  Is the container available to receive a command or request?

- boolean isContainerAvailable (ISAOSpec spec)

  Is the container available to receive a command or request?

- short getContainer (ISAOSpec spec)

  Determine the container of an ISAOSpec.

- int getContainerState (short cid)

  Determine the availability of a container.

- int getUserId ()

  Get the user Id of the user that created this client.

- String getApplicationName ()

  Returns the name of the application being used by this client.

- String getUserName ()

  The name of the user this client was successfuly initialised with.

- String getAppName ()

  The name of the application this instance of ISAFClient is connected to.

- Object getMonitor ()

  Gets the monitor object to be used to lock the client instance from the non S/AF thread in a synchronized code block.

- boolean requestTable (String name, short cid, long requestId, IUpdatingTableListener tableListener)

  Request a table.

- boolean setConfiguration (long requestId, String name, ISAFValue value, int unit, ISAOListener saoListener)

  Change or add a "variable" in the IUpdatingTable#CONFIGURATION_TABLE.

- boolean resetConfiguration (long requestId, String name, int unit, ISAOListener saoListener)

  Remove a "variable" in the IUpdatingTable#CONFIGURATION_TABLE.

- boolean requestRecord (String name, long requestId, ISAOListener saoListener)

  Request data conforming to the record interface.

- boolean addMachine (String machine, long rid, ISAOListener saoListener)

  Add a machine to those defined in the application.

- boolean removeMachine (String machine, long rid, ISAOListener saoListener)

  Remove a machine from the application.

- boolean addProcess (String machine, String process, String dbdllName, String dbstring, String password, long rid, ISAOListener saoListener)

  Add a process to those defined in the application.

- boolean addProcess (String machine, String process, long rid, ISAOListener saoListener)

  Add a process to those defined in the application.

- boolean removeProcess (String machine, String process, long rid, ISAOListener saoListener)

  Remove a process from the application.

- boolean addContainer (String machine, String process, String container, String priority, long rid, ISAOListener saoListener)

  Add a container to a process.

- boolean removeContainer (String machine, String process, String container, long rid, ISAOListener saoListener)

Remove a container from a process.

- boolean renameSAO (ISAOSpec spec, String name, long rid, ISAOListener saoListener)

  Rename an SAO.

- boolean applicationControl (long requestId, int type, long timeout, ISAOListener listener)

  Stop/Start/Exit an application.

- boolean exportHierarchy (ISAOSpec root, long requestId, int control, String description, int scope, ISAOListener listener)

  Export a part of an applications hierarchy into XML.

- boolean exportHierarchy (ISAOSpec root, long requestId, String file, int control, String description, int scope, ISAOListener listener)

  Export a part of an applications hierarchy and write the generated XML into the supplied file.

- boolean importHierarchy (ISAOSpec target, long requestId, byte[ ] exportXML, String name, int loc, ISAOSpec next, int control, int scope, ISAOListener listener)

  Import hierarchy from XML.

- boolean importHierarchy (ISAOSpec target, long requestId, String fileName, String name, int loc, ISAOSpec next, int control, int scope, ISAOListener listener)

  Import hierarchy from a file containing XML.

A.2.2 Member Function Documentation

int com.orimos.saf.client.ISAFClient.initialise () `[package]`

Initialise this instance.

This must be called before any other method except possibly setTimeout. It is intended to be used to initialise client instances that have been created with SAFClient.createClient(). If it is called on a Swing client instance the method with throw an UnsupportedOperationException

Returns:

The method can return the following codes If it returns RETRY then it should be repeated after a delay.
If it returns OK you are successfully initialized.
If it returns NOSONARISMESSAGING this means that if could not connect to the local messaging service - you could exit or treat this as it it returned RETRY and it will connect after the messaging is started.
If it returns NOSAFAPPLICATION then this implies that it has reached the timeout time and the data being fetched from the application has not been completely retrieved, most likely because it isn't up. Again you can reset the timeout and continue.

If it returns NOSAFLOGIN then the user/password supplied were not accepted - you may use newCredentials and call initialise again or you could exit the program.

int com.orimos.saf.client.ISAFClient.initialiseSwingClient () `[package]`

Initialise this instance.

This must be called before any other method except possibly setTimeout. You must *not* call this method on the Swing event thread as initialisation uses that thread, and it must not be blocked. Generally, it is best to call the method on the applications 'main' thread. The method will check that the client being initialised is a swing client. If not an UnsupportedOperationException will be thrown.

Returns:

The method can return the following codes If it returns RETRY then it should be repeated after a delay.
If it returns OK you are successfully initialized.
If it returns NOSONARISMESSAGING this means that if could not connect to the local messaging service - you could exit or treat this as it it returned RETRY and it will connect after the messaging is started.
If it returns NOSAFAPPLICATION then this implies that it has reached the timeout time and the data being fetched from the application has not been completely retrieved, most likely because it isn't up. Again you can reset the timeout and continue.
If it returns NOSAFLOGIN then the user/password supplied were not accepted - you may use newCredentials and call initialise again or you could exit the program.

void com.orimos.saf.client.ISAFClient.setTimeout (int *secs*) `[package]`

Sets a timeout for initialise.

The default timeout is 10 seconds after creation of the client. The timeout, if needed, is sheduled for this number of seconds after the time of this call. If you call initialise after the scheduled timeout time it could timeout immediately. Setting the value to zero will prevent initialise from timing out.

Parameters:

secs The timeout

long com.orimos.saf.client.ISAFClient.getRid () `[package]`

Provide a request Id.

Currently this method increments an internal long. There is no requirement to use this method to provide a request Id. However, this method is safe and ensures that there will not be re-use of request ids from the same client instance.

Returns:

A request Id.

boolean com.orimos.saf.client.ISAFClient.hasLicense () `[package]`

Returns:

true if this client has access to a legitimate license

int com.orimos.saf.client.ISAFClient.terminate () `[package]`

Terminate the client.

This removes the SAF connection and tidies up memory used to manage the connection for the client. If requests are still active when this method is called, each will be terminated by a cancelled callback; there will then be a terminated callback on the associated ISAFClientListener. After this the SAFClient instance is deleted and will not be usable or capable of reinitialisation. The results of any uncompleted commands are indeterminate.

Returns:

The return code can currently be ignored

void com.orimos.saf.client.ISAFClient.lock () `[package]`

Locks in preparation for accessing ISAOs.

ISAOs should not be looked for or accessed outside the callback methods without taking a lock first. This is because any part of an SAO might change at any time due to actions taking place on another thread.

ISAO com.orimos.saf.client.ISAFClient.find (ISAOSpec *id*) `[package]`

Finds an SAO locally.

Will only find an SAO if it has been obtained directly or indirectly by some request. Only to be used after taking a lock.

Parameters:

id The Id of the required SAO.

Returns:

null if the SAO is not present (it may not be present even if it has been requested as the response for the request may not have arrived)

ISAO com.orimos.saf.client.ISAFClient.find (String *name*) `[package]`

Finds an SAO.

Will only find an SAO if it has been obtained directly or indirectly by some request. Will only find an SAO if it, or one of its direct or indirect parents, has been requested by name. Only to be used after taking a lock.

Parameters:

name The name of the required SAO.

Returns:

null if the SAO is not present (it may not be present even if it has been requested as the response for the request may not have arrived)

ISAO com.orimos.saf.client.ISAFClient.find (ISAO *base*, String *name*) `[package]`

Finds an SAO by name relative from a given SAO.

Will only find an SAO if it has been obtained directly or indirectly by some request. Only to be used after taking a lock.

Parameters:

base The SAO to start searching in. The first component of the name is expected to be one of this SAOs children.

name The name of the required SAO.

Returns:

null if the SAO is not present (it may not be present even if it has been requested as the response for the request may not have arrived)

void com.orimos.saf.client.ISAFClient.cancel (long *requestId*) `[package]`

Cancel a request for information from an SAO.

You will continue to receive responses for this requestId until you get a cancelled() callback. It is not safe to re-use the requestId until then. You will get a cancelled callback quoting the requestID even if the requestId is not recognised.

Parameters:

requestId The Id to be cancelled

boolean com.orimos.saf.client.ISAFClient.request (ISAOSpec *id*, long *requestId*, boolean *data*, boolean *includeChildren*, boolean *snapshot*, ISAOListener *saoListener*, ISAOPropertyListener *propListener*) `[package]`

Request an SAO.

The SAO is requested and the call returns at once. The caller will be notified of data availability by a call from the ISAOListener interface. Requesting a non-existent SAO by ID will result in a single failed callback. <p/> The caller can request the SAO data or its children or both.

Parameters:

id Identifies the target.

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

data Should the request supply the data of the SAO.

includeChildren Should the request include the children of the SAO.

snapshot If this is true then you will only get only one response followed by a cancelled callback. If this is false you will get a response followed by updates when the SAO changes.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

propListener If non NULL then this request includes extended information and this address is used to report changes.

Returns:
true if it works and false if it fails.

Exceptions:
IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.activate (ISAO *sao*, long *requestId*, boolean *data*, boolean *includeChildren*, boolean *snapshot*, ISAOListener *saoListener*, ISAOPropertyListener *propListener*) `[package]`

Activate an SAO.

An SAO can be activated if it has been obtained from the framework by requesting its parent. This is equivalent to a request() by SAOId. Note that you only normally have access to SAOs from within callbacks (or after taking a lock).

Parameters:

sao The SAO to activate

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request. If you want to upgrade the subscription on an existing request then you will have to issue a new request with a different ID.

data Should the request supply the data of the SAO.

includeChildren Should the request include the children of the SAO.

snapshot If this is true then you will only get only one response followed by a cancelled callback. If this is false you will get a response followed by updates when the SAO changes.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

propListener If non NULL then this request includes extended information and this address is used to report changes.

Returns:
true if it works and false if it fails.

Exceptions:
IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.modify (long *requestId*, ISAOPropertyListener *propListener*) `[package]`

Change the ISAOPropertyListener associated with a request.

This method must only be called from within a callback.

Parameters:

requestId The unique identifier of an existing request.

propListener If NULL then property update calbacks are switched off; if the same as the previous listener associated with the request then updates are neither switch on or off and this call does nothing; otherwise property updates are switched on and the new address used for all subsequent callbacks. Switching properties on will generate a response callback when all the details are fetched ; this may occur before return is made from this call if the details are already available.

Returns:

true if it works and false if it fails.

boolean com.orimos.saf.client.ISAFClient.setSAOValue (long *requestId*, ISAOSpec *id*, ISAFValue *value*, ISAOListener *saoListener*) `[package]`

Use this method to change the value of an SAO.

Note that no compatibility checking can be performed prior to sending the message and that SAOs are not guaranteed to acccept the value.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

id Identifies the SAO to receive the value.

value The value it is to be given.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. Re-using a requestId will result in the original request being cancelled (you will get a cancelled callback) and the new request replacing it.

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.setSAOValue (long *requestId*, ISAOSpec *id*, ISAFValue *value*, ISAOListener *saoListener*, boolean *complete*) `[package]`

Use this method to change the value of an SAO.

Note that no compatibility checking can be performed prior to sending the message and that SAOs are not guaranteed to acccept the value.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

id Identifies the SAO to receive the value.

value The value it is to be given.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

complete This allows a set of updates to more than one SAO to be collected together and delivered as one message and processed as a group. To use this facility you should make a series of calls, each with the same requestId, with complete set to false on all but the last. The full set will be collected and processed with this final entry. If the SAOs to be updated are in different containers then the set will fail via a failed() callback. If not all the SAOs could take the given value then none of them will be written and you will get a failed() callback. The saoListener value is only used if complete is true.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. Re-using a requestId will result in the original request being cancelled (you will get a cancelled callback) and the new request replacing it.

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.copyTree (long *requestId*, ISAOSpec *source*, String *name*, ISAOSpec *parent*, ISAOSpec *next*, boolean *all*, boolean *transientCopy*, ISAOListener *saoListener*) `[package]`

Use this method to copy a tree of SAOs.

Inputs are duplicated in the new tree.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

source Identifies the SAO at the root of the tree to be copied.

name The name of the new SAO. May not be NULL. This command will fail if there is already a child of this name in the parent.

parent Identifies the parent of the new child. Can be NULL in which case the new SAO goes at the root of the container containing the source. If specified it must be in the same

next Defines the position of new child in the parent. If NULL then the child goes on the end of the list of the parent's children. If specified this command will fail unless the identified sibling exists and is a child of the same parent, otherwise the new child will be placed before the given one.

all If true then all SAOs in the source tree are copied; if false then only persistent SAOs are copied.

transientCopy If true then all new SAOs are made transient. This will be necessary if the parent is itself a transient SAO.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. The commandResult will be the SAOId of the newly created SAO on success.

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.moveTree (long *requestId*, ISAOSpec *source*, ISAOSpec *parent*, ISAOSpec *next*, ISAOListener *saoListener*) `[package]`

Use this method to move a tree of SAOs.

Inputs are duplicated in the new tree. This is much preferable to copy and delete when the source and destination are in the same container and amounts to the same thing when they are not.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

source Identifies the SAO at the root of the tree to be moved.

parent Identifies the new parent. Can be NULL in which case the current parent of the SAO is used. The command will fail if the parent already has a child (other than source) with the source's name.

next Defines the position of new child in the parent. If NULL then the child goes on the end of the list of the parent's children. If specified this command will fail unless the identified sibling exists and is a child of the same parent, otherwise the SAO will be placed before the given one.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. The commandResult will be the SAOId of the destination SAO on success (usually this is the id or the source SAO).

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.modifyTree (long *requestId*, ISAOSpec *source*, ISAOSpec *destination*, ISAOListener *saoListener*, int *control*) `[package]`

Use this method to modify a tree of SAOs to match a template.

The destination will be transformed by adding and removing SAOs and adding and removing inputs so as to match the source.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

source Identifies the SAO at the root of the tree to be used as the template.

destination Identifies the SAO which will be modified to match the source.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

control Contains a combination of bits from ISAFValue.MTC_TRANSIENT and/or ISAFValue.MTC_CHILDREN and is used to control the operation. See ISAFValue.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. The commandResult will be the SAOId of the destination SAO on success (usually this is the id or the source SAO) and zero on failure.

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.invokeCommandSAO (ISAOSpec *id*, long *requestId*, ISAFProperties *params*, ISAOListener *saoListener*) `[package]`

This method invokes the remoteCommand method of an SAO.

You should use it with care because there is no type checking on the parameters supplied. You will only get one response callback, either a failed on failure or a succeeded on success.

Parameters:

id Identifies the target.

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

params The data to be sent to the SAO. Ownership is taken of this object and it will be deleted later. It may be NULL.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback.

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.addSAO (long *requestId*, SAOFType *type*, String *name*, ISAOSpec *parent*, ISAOSpec *next*, boolean *persistent*, ISAOListener *saoListener*) `[package]`

Create and insert an SAO by functional type.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

type The type of SAO to create.

name The name of the new SAO. May not be NULL or empty. This command will fail if there is already a child of this name in the parent.

parent Identifies the parent of the new SAO. Can be a full path or an Id. If an ID the objectId may be zero if it is to be placed at the top of the container but the containerId must not be zero.

next Defines the position of new child in the parent. If not present then the child goes on the end of the list of the parent’s children. If specified this command will fail unless the identified sibling exists and is a child of the same parent, otherwise the new child will be placed before the given one.

persistent If false then the newly created SAO is not persisted to the database.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. The commandResult will be the SAOId of the newly created SAO on success.

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.addSAO (long *requestId*, String *libName*, String *saoName*, String *name*, ISAOSpec *parent*, ISAOSpec *next*, boolean *persistent*, ISAOListener *saoListener*) `[package]`

Create and insert an SAO by name.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

libName The name of the SAO library containing the SAO definition.

saoName The name of the SAO definition within the library.

name The name of the new SAO. May not be NULL. This command will fail if there is already a child of this name in the parent.

parent Identifies the parent of the new SAO. Can be a full path or an Id. If an ID the objectId may be zero if it is to be placed at the top of the container but the containerId must not be.

next Defines the position of new child in the parent. If not present then the child goes on the end of the list of the parent's children. If specified this command will fail unless the identified sibling exists and is a child of the same parent, otherwise the new child will be placed before the given one.

persistent If false then the newly created SAO is not persisted to the database.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. The commandResult will be the SAOId of the newly created SAO on success.

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.removeSAO (long *requestId*, ISAOSpec *sao*, ISAOListener *saoListener*) `[package]`

Use this method to remove an SAO.

Refer to the relevant Command SAO's documentation.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

sao Identifies the SAO to be removed.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback.

boolean com.orimos.saf.client.ISAFClient.linkSAO (long *requestId*, ISAOSpec *destination*, ISAOSpec *source*, int *index*, boolean *overwrite*, ISAOListener *saoListener*) `[package]`

Use this method to create a dependancy between two SAOs.

Refer to the relevant Command SAO's documentation. Note that the source and destination might be in different containers.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identifiy the request.

destination Identifies the SAO which will have in input set up.

source Identifies the SAO which will have an output set up.

index Input number in the destination. If -1 then the new SAO will go in at any suitable index.

overwrite If true then an existing input at the index in the destination will be trashed.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. Re-using a requestId will result in the original request being cancelled (you will get a cancelled callback) and the new request replacing it. The commandResult will be the input number chosen if the index was given as -1 on success.

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

boolean com.orimos.saf.client.ISAFClient.unlinkSAO (long *requestId*, ISAOSpec *destination*, ISAOSpec *source*, int *index*, ISAOListener *saoListener*) `[package]`

Use this method to remove a dependancy between two SAOs.

Refer to the relevant Command SAO's documentation. Note that the source and destination might be in different containers.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identifiy the request.

destination Identifies the SAO which will have an input removed.

source Identifies the SAO which is an input and which should be removed. May be NULL.

index Input number in the destination. If not -1 then this number input is removed (and the source parameter is completely ignored).

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. Re-using a requestId will result in the original request being cancelled (you will get a cancelled callback) and the new request replacing it.

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

ISAFPermissioning com.orimos.saf.client.ISAFClient.getPermissioning () `[package]`

Provides access to the permissioning information.

Note that any information returned from the ISAFPermissioning object is only safe to use from within a callback or with the lock taken because it can asynchronously change in the callback thread. Returns null if there is no information available. There will be none available if you did not request this on the first call of createClient.

boolean com.orimos.saf.client.ISAFClient.changePassword (long *requestId*, String *doer*, String *oldpassword*, String *user*, String *newpassword*, ISAOListener *saoListener*) `[package]`

Change a user’s password.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

doer The name of the user changing the password. This must be the same name supplied when the ISAFClient was created.

oldpassword The existing password of the doer

user The name of the user having the passsword changed. This will usually be the same as the doer - if not then you must be logged on as a superuser.

newpassword The password to be set for the user.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. The commandResult will be the SAOId of the destination SAO on success (usually this is the id or the source SAO).

Exceptions:

IllegalArgumentException if the saoListener is null and no global saoListener has been supplied

void com.orimos.saf.client.ISAFClient.encryptPassword (String *user*, String *pw*, ISAFValue *output*) `[package]`

Encrypts a username and password into an ISAFValue; special function to be used when changing a user's password via a command SAO.

Parameters:

user The user name

pw The associated plain text password. May be NULL if the user has no need to supply a password.

output Somewhere to put the resultant string.

boolean com.orimos.saf.client.ISAFClient.newCredentials (String *user*, String *pw*) `[package]`

Replace the username and password supplied to SAFClient#createClient SAFClient#createSwingClient or previous calls to this method with those specified.

If initialise or initialiseSwingClient returns SAFClient#NO_SAF_LOGIN then this method can be used before retrying initialise(); calls to this method made after the ISAFClient is successfully initialised have no effect and return false.

Note: A single client program may make several calls to SAFClient#createClient in order to create multiple ISAFClient instance with which it can access multiple different Framework application services. Each such call may supply different credentials as appropriate to the application service. It is also possible to create multiple ISAFClient instances which access the same application service. In this case, however, the credentials of the first ISAFClient created will be used when initialising *all* other ISAFClient instances which connect to that application service.

Parameters:

user The username.

pw The associated plain text password. May be null if no password is required to authenticate this user.

Returns:

True if the credentials have been accepted, false if not.

boolean com.orimos.saf.client.ISAFClient.verifyCredentials (String *user*, String *pw*) `[package]`

Determine if a user name and password are valid and whether or not the user has system privilege.

Parameters:

user The username

pw The associated plain text password. May be NULL if no password is required to authenticate this user.

Returns:

- null if the ISAFClient on which this call is being made has not yet been successfully initialized.
- null if the credentials are not recognised.
- An ISAFUser instance representing the client.

void com.orimos.saf.client.ISAFClient.cancelSAOId (SAOId *saoId*) `[package]`

Cancel request for information from an SAO by SAOId.

This cancels all requests (0, 1 or more) that reference this SAO. It leads to the same number of ISAOListener::cancelled() callbacks as there were requests that were cancelled. You will continue receive responses for an affected requestId until you get a cancelled() callback for the requestId. It is not safe to re-use the requestId until then.

Parameters:

saoId The id of the SAO whose requests are to be cancelled

boolean com.orimos.saf.client.ISAFClient.reserveSAO (long *requestId*, ISAOSpec *dest*, ISAOListener *saoListener*) `[package]`

Use this method to mark an SAO at the server.

Reserving an SAO will allow applications to cooperate as an SAO can only be reserved once. Reserving an SAO has no effect other than that another call to reserveSAO will fail. A reservation will end either by the reserver calling releaseSAO or when this client process terminates.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

dest Identifies the SAO to be reserved.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:

true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. The commandResult will be the SAOId of the reserved SAO on success and zero on failure; the command will fail if the SAO is already reserved.

boolean com.orimos.saf.client.ISAFClient.releaseSAO (long *requestId*, ISAOSpec *dest*, ISAOListener *saoListener*) `[package]`

Use this method to release a reserved SAO at the server.

Parameters:

requestId A unique identifier supplied with the request. This identifier is returned with callbacks so that the client can identify the request.

dest Identifies the SAO to be released.

saoListener If non NULL the callback address to be used for responses; if NULL then there must have been a default provided on createClient.

Returns:
true if it works and false if it fails. Normally this call will complete asynchonously with a later succeeded or failed callback. The commandResult will be 1 on success and zero on failure.

void com.orimos.saf.client.ISAFClient.addClientListener (ISAFClientListener *listener*) `[package]`

Add a new client listener to the current set.

Parameters:
listener The listener

void com.orimos.saf.client.ISAFClient.removeClientListener (ISAFClientListener *listener*) `[package]`

Remove a client listener from this client.

Parameters:
listener The listener

int com.orimos.saf.client.ISAFClient.getContainerState (ISAOSpec *spec*) `[package]`

Determine the availability of a container.

Parameters:
spec Used to specify indirectly the container which is being reported. This operation internally locks the client.

Returns:
The state of the container. If the container cannot be determined then SAF_NOTEXIST is returned.

boolean com.orimos.saf.client.ISAFClient.isContainerAvailable (short *cid*) `[package]`

Is the container available to receive a command or request?

Parameters:
cid The container Id.

Returns:
true if the container is available.

boolean com.orimos.saf.client.ISAFClient.isContainerAvailable (ISAOSpec *spec*) `[package]`

Is the container available to receive a command or request?

Parameters:

spec The ISAOSpec used to determine the container to be checked.

Returns:

true if the container is available.

short com.orimos.saf.client.ISAFClient.getContainer (ISAOSpec *spec*) `[package]`

Determine the container of an ISAOSpec.

Parameters:

spec Used to specify indirectly the container which is being reported. This operation internally locks the client.

Returns:

The container. If the container cannot be determined then SAF_NOCONTAINER is returned.

int com.orimos.saf.client.ISAFClient.getContainerState (short *cid*) `[package]`

Determine the availability of a container.

This operation internally locks the client.

Parameters:

cid A container number or SAF_NOCONTAINER if you wish to know about the overall availability of the application. See ISAFClientStatus::ContainerState_t.

Returns:

The state of the container. If the container cannot be determined then SAF_NOTEXIST is returned.

int com.orimos.saf.client.ISAFClient.getUserId () `[package]`

Get the user Id of the user that created this client.

Returns:

The user Id or ISAFPermissioning#NO_USERID if the client is not connected to an application or did not ask for permissioning information when the ISAFClient was created.

String com.orimos.saf.client.ISAFClient.getUserName () `[package]`

The name of the user this client was successfuly initialised with.

If the method is called prior to initialisation being complete or on a client that failed initialisation it will return the user name specifed on SAFClient#createClient orSAFClient#createSwingClient. Calls to newCredentials after initialisation has completed do not affect the user name.

Returns:
The user name.

String com.orimos.saf.client.ISAFClient.getAppName () `[package]`

The name of the application this instance of ISAFClient is connected to.

Returns:
The name of the application or null if an application is not connected.

Object com.orimos.saf.client.ISAFClient.getMonitor () `[package]`

Gets the monitor object to be used to lock the client instance from the non S/AF thread in a synchronized code block.

Synchronized blocks are preferable to calling lock() and unlock() as there's no need to make sure that there are pairs of calls.

boolean com.orimos.saf.client.ISAFClient.requestTable (String *name*, short *cid*, long *requestId*, IUpdatingTableListener *tableListener*) `[package]`

Request a table.

Responses, updates and other notifications are supplied through the table listener.

Parameters:

name The name of the table.

cid The container Id of the container running the table. Unless specifically stated otherwise, this should be 0.

requestId A request Id to use with the request.

tableListener A listener to receive notifications of responses, updates etc.

Returns:
true if the request was made.

void com.orimos.saf.client.ISAFClient.cancelTable (long *requestId*)

Cancel a table request.

Parameters:

requestId The request Id of the request to cancel.

boolean com.orimos.saf.client.ISAFClient.setConfiguration (long *requestId*, String *name*, ISAFValue *value*, int *unit*, ISAOListener *saoListener*) `[package]`

Change or add a ”variable” in the IUpdatingTable#CONFIGURATION_TABLE.

The setting will be persisted to the database and be retained after a restart. Only one ISAOListener response callback is made, either ISAOListener#succeeded on success or ISAOListener#failed on failure.

Parameters:

value The value of the variable. Currently only string and int values are supported.

name The name of the variable. May not be NULL or ””. Case is sensitive.

unit Unit number if applicable, or -1 if not. The unit number for container configuration is the container id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.resetConfiguration (long *requestId*, String *name*, int *unit*, ISAOListener *saoListener*) `[package]`

Remove a ”variable” in the IUpdatingTable#CONFIGURATION_TABLE.

The setting will be persisted to the database and be retained after a restart. Only one ISAOListener response callback is made, either ISAOListener#succeeded on success or ISAOListener#failed on failure.

Parameters:

name The name of the variable. May not be NULL or ””. Case is sensitive.

unit Unit number if applicable, or -1 if not. The unit number for container configuration is the container id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.requestRecord (String *name*, long *requestId*, ISAOListener *saoListener*) `[package]`

Request data conforming to the record interface.

If there is a record associated with the specified SAO then it will be delivered by ISAOListener#response and ISAOListener#update callbacks; if no record is present then this will be indicated via a ISAOListener#failed callback. Note that there may be any combination of response and update callbacks, and some responses may have a SAFStatus#SAF_INVALID or SAFStatus#SAF_FAILOVER status. <p/> This command is available without a login to SAF; i.e. you do not need to initialize the ISAFClient to use it.

Parameters:

name The name of the record to be fetched. TRequest will fail immediately if this is NUll or empty.

requestId A unique identifier supplied with the request. This identifier is returned with ISAOListener callbacks so that the client can identify the request.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.addMachine (String *machine*, long *rid*, ISAOListener *saoListener*) `[package]`

Add a machine to those defined in the application.

Parameters:

machine The name of the machine.

rid The command Id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.removeMachine (String *machine*, long *rid*, ISAOListener *saoListener*) `[package]`

Remove a machine from the application.

If the machine has processes and containers defined, they will also be removed.

Parameters:

machine The name of the machine.

rid The command Id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.addProcess (String *machine*, String *process*, String *dbdllName*, String *dbstring*, String *password*, long *rid*, ISAOListener *saoListener*) `[package]`

Add a process to those defined in the application.

The machine must exist and the process name must be for that machine.

Parameters:

machine The name of the machine.

process The name of the process.

dbdllName The name of the database dll.

dbstring The database string.

password The database password.

rid The command Id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.addProcess (String *machine*, String *process*, long *rid*, ISAOListener *saoListener*) `[package]`

Add a process to those defined in the application.

The machine must exist and the process name must be for that machine.

Parameters:

machine The name of the machine.

process The name of the process.

rid The command Id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.removeProcess (String *machine*, String *process*, long *rid*, ISAOListener *saoListener*) `[package]`

Remove a process from the application.

If the processes has containers defined they will also be removed.

Parameters:

machine The name of the machine.

process The name of the process.

rid The command Id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.addContainer (String *machine*, String *process*, String *container*, String *priority*, long *rid*, ISAOListener *saoListener*) `[package]`

Add a container to a process.

The machine and process must already exist. If the process has not been disabled and this is the first instance of this container, the container will start running.

Parameters:

machine The name of the machine.

process The name of the process.

container The name of the container.

priority The priority of the container. One of

- IUpdatingTable#LTCONTAINERS_HIGH High priority.
- IUpdatingTable#LTCONTAINERS_MEDIUM_1 Medium priority.
- IUpdatingTable#LTCONTAINERS_LOW Low priority.

rid The command Id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.removeContainer (String *machine*, String *process*, String *container*, long *rid*, ISAOListener *saoListener*) `[package]`

Remove a container from a process.

If this is the only instance of the container it will be stopped. If the container is the running instance and there are standby instances, the running instance will be stopped and a standby instance started.

Parameters:

machine The name of the machine.

process The name of the process.

container The name of the container.

rid The command Id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.renameSAO (ISAOSpec *spec*, String *name*, long *rid*, ISAOListener *saoListener*) [package]

Rename an SAO.

Some SAOs may refuse to have their name changed, in which case the command will fail.

Parameters:

spec The ISAOSpec identifying the SAO to have it's name changed.

name The new name for the SAO.

rid The command Id.

saoListener An implementation of the ISAOListener interface to be used for responses. If null then defaults to using the implementation supplied during the call to createClient().

Returns:

False if the call is rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.applicationControl (long *requestId*, int *type*, long *timeout*, ISAOListener *listener*) [package]

Stop/Start/Exit an application.

You must have admin level access to the application for this command to work.

Parameters:

requestId The request Id.

type The type of operation.

- 0 is STOP
- 1 is START
- 2 is EXIT

timeout The time to wait before the application stops or exits, -1 applies is no timeout.

listener The listener called back to when the command completes.

Returns:

true if the command was sent, false otherwise.

boolean com.orimos.saf.client.ISAFClient.exportHierarchy (ISAOSpec *root*, long *requestId*, int *control*, String *description*, int *scope*, ISAOListener *listener*) [package]

Export a part of an applications hierarchy into XML.

The resulting export file contains everything needed to recreate the exported hierarchy via import-Hierarchy, there are options to control what is exported, but, generally it is as well to export everything and select what to import. The command operates by looking at the root of the export and determining how many containers are involved in the export. Sub-commands are then sent to each container to gather the basic export information.

If the export asked for external links to be preserved the sub-command results are examined for external link information and each externally linked SAO has its full path resolved by sending further sub-commands to the relevant containers.

Once the external links have been gathered the sub-command results are merged and the export XML is generated.

This command differs from others in ISAFClient in that it returns progress information via ISAOListener#update. The method is called at various stages during the export with an IResult#getCommandResult() that has a SAFValueType#SAF_STRING_ARRAY with the following content:

- Index 0 A message relating to the stage the export is at.
- Index 1 An integer indicating the number of completed steps
-
- Index 2 An integer indicating the total number of steps. This number may increase as the command progresses and it determines that more work is required.

Subsequent elements are as yet undefined. These callbacks are intended for UIs where some form of feedback on the progress of extended commands is advisable. Messages should be treated as being opaque and not fit for parsing.

When the command is complete the ISAOListener#succeeded callback will have a command result data containing a SAF_STRING that has the export XML.

This command is not intended to be used as a mechanism for archiving and entire application. Generating XML to describe several hundred thousand SAOs, their values and inputs will require significant memory both to transport information from the containers and then to create the XML. Exports that span a lot of containers are also likely to generate significant work for those containers both in supplying the SAO information for the export and resolving external link information. This should be born in mind when running an export.

Parameters:

root The point in the hierarchy where the export will start.

requestId The Id of the command.

control Parameters controlling what is exported.

description A Description of the content of the export.

scope Determines the way in which the command deals with unavailable containers.

listener The listener to be used to notify of the success or failure of the command and the commands progress. If NULL the listener supplied when the client was created will be used.

Returns:

false if the command was rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.exportHierarchy (ISAOSpec *root*, long *requestId*, String *file*, int *control*, String *description*, int *scope*, ISAOListener *listener*) `[package]`

Export a part of an applications hierarchy and write the generated XML into the supplied file.

This method will use less memory and probably be slightly quicker to execute than first using the method that returns the XML in a buffer and then writing that output to a file.

The specified file will be created or overwritten.

Parameters:

root The point in the hierarchy where the export will start.

requestId The Id of the command.

file The full path of a file to contain the exported XML.

control Parameters controlling what is exported.

description A description of the the content of the export file.

scope Determines the way in which the command deals with unavailable containers.

listener The listener to be used to notify of the success or failure of the command and the commands progress. If NULL the listener supplied when the client was created will be used.

Returns:

false if the command was rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.importHierarchy (ISAOSpec *target*, long *requestId*, byte[ ] *exportXML*, String *name*, int *loc*, ISAOSpec *next*, int *control*, int *scope*, ISAOListener *listener*) `[package]`

Import hierarchy from XML.

If external links are to be made to the imported hierarchy the command will check that they are present and the command will fail unless the IGNORE_MISMATCHED_INPUTS flag is set. If the command fails for this reason the failed links will be returned in the command result as a SAFValueType#SAF_STRING_ARRAY, one failed link in each array element.

When the command is complete the ISAOListener#succeeded callback will have a command result data containing a SAF_INT64 that is the SAOId of the root of the newly created hierarchy.

Imports can take a significant time to complete if they require the creation of large numbers of persistent SAOs, the equivalent of copying a large tree from one container to another. When a container is digesting a large import it will not be doing anything else, so it is important to be sensitive to the performance implications of importing SAOs

This command differs from others in ISAFClient in that it returns progress information via ISAOListener#update. The method is called at various stages during the export with an IResult#getCommandResult() that has a SAFValueType#SAF_STRING_ARRAY with the following content:

- Index 0 A message relating to the stage the export is at.

- Index 1 An integer indicating the number of completed steps
-
- Index 2 An integer indicating the total number of steps. This number may increase as the command progresses and it determines that more work is required.

Subsequent elements are as yet undefined. These callbacks are intended for UIs where some form of feedback on the progress of extended commands is advisable. Messages should be treated as being opaque and not fit for parsing.

When the command is complete the ISAOListener#succeeded callback will have a command result data containing a SAF_STRING that has the export XML.

Parameters:

target The parent of the SAOs to be imported.

requestId The Id of the command.

exportXML The XML containing details of SAOs to be imported.

name The name of the root of the imported hierarchy. If null this will be the root name from the export file.

loc The location of the SAO relative to the target.

next The next sibling for the imported tree. If NULL the tree will be the last child of the parent.

control Parameters to control the operation of the import.

scope Controls the behaviour of the import if containers for external inputs are not available.

listener The listener to be used to notify of the success or failure of the command and the commands progress. If NULL the listener supplied when the client was created will be used.

Returns:

false if the command was rejected immediately, true otherwise.

boolean com.orimos.saf.client.ISAFClient.importHierarchy (ISAOSpec *target*, long *requestId*, String *fileName*, String *name*, int *loc*, ISAOSpec *next*, int *control*, int *scope*, ISAOListener *listener*) `[package]`

Import hierarchy from a file containing XML.

This is more convenient and will use less memory than the buffer based method in situations where the XML has to be read from a file. In all other respects this method has the same requirements and behaviour as the buffer based equivalent.

Parameters:

target The parent of the SAOs to be imported.

requestId The Id of the command.

fileName The file containing XML.

name The name of the root of the imported hierarchy. If null this will be the root name from the export file.

loc The location of the SAO relative to the target.

next The next sibling for the imported tree. If NULL the tree will be the last child of the parent.

control Parameters to control the operation of the import.

scope Controls the behaviour of the import if containers for external inputs are not available.

listener The listener to be used to notify of the success or failure of the command and the commands progress. If NULL the listener supplied when the client was created will be used.

Returns:
false if the command was rejected immediately, true otherwise.

A.2.3 Member Data Documentation

final long com.orimos.saf.client.ISAFClient.NO_REQUEST = Long.MIN_VALUE `[static]`

A request Id which can never be obtained from this client unless more than Long.MAX_VALUE requests are made.

Normally, this would be Long.MIN_VALUE, but for debugging purposes it's easier to read numbers when they're low positive

Definition at line 211 of file ISAFClient.java.

final String com.orimos.saf.client.ISAFClient.SAFVARS_PROPERTY = PROPERTY_NAME_PATH + "saf.safvars" `[static]`

The properties key for the safvars properties object.

This is a standard java.util.Properties and should be used as such. It contains properties useful (but not essential to) SAF client applications.

Definition at line 233 of file ISAFClient.java.

final String com.orimos.saf.client.ISAFClient.SAFVARS_LOCATION_PROPERTY = PROPERTY_NAME_PATH + "safvars" `[static]`

The property path of the system properties property that can be used to define a new location for the SAF properties file.

The value is expected to be a path to a directory (with or without trailing separator)

Definition at line 242 of file ISAFClient.java.

final String com.orimos.saf.client.ISAFClient.SAFDIR_PROPERTY = PROPERTY_NAME_PATH + "safdir" `[static]`

The directory into which the current SAF installation has been installed.

This is normally part of the root of BIN_DIR_PROPERTY but need not be so...

Definition at line 251 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.STOP = 1 `[static]`

Flag to stop an application.

Stopping it leaves safbase running, but stops all the saf processes.

See also:
applicationControl

Definition at line 1109 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.START = 0 `[static]`

Flag to start an application.

The app must have previously been in the 'stopped' state and this will start all the saf processes.

See also:
applicationControl

Definition at line 1116 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.EXIT = 2 `[static]`

Flag to exit an application.

Exiting stops all the saf processes and then safbase.

See also:
applicationControl

Definition at line 1122 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.INCLUDE_INPUTS = 0x1 `[static]`

These constants control the content of exported XML or an imported hierarchy.

- **INCLUDE_INPUTS**Make inputs between SAOs that are part of the exported SAOs. If there is no input information in the export this option has no effect.
- **INCLUDE_VALUES**SAOs imported have their values set from values in the export data. If there no values in the export this option has no effect.
- **INCLUDE_TRANSIENTS**Tansient SAOs are imported from the export data. If there no values in the export this option has no effect.

- **INCLUDE_EXTERNAL_INPUTS**SAOs imported have links made to SAOs outside the imported hierarchy. If there is no external link information in the export this option has no effect.

Definition at line 1147 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.NORMAL_EXPORT `[static]`

Initial value:

```
INCLUDE_VALUES | INCLUDE_INPUTS |
        INCLUDE_EXTERNAL_INPUTS | INCLUDE_PERMISSIONS
```

These are the settings of the control parameters for most exports.

Definition at line 1158 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.NORMAL_IMPORT `[static]`

Initial value:

```
INCLUDE_VALUES | INCLUDE_INPUTS |
        INCLUDE_EXTERNAL_INPUTS | INCLUDE_PERMISSIONS
```

This defines an import with the most useful control parameters set.

Definition at line 1163 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.SEND_AND_WAIT = 0 `[static]`

Description: Controls the behaviour of a command when it has to send sub-commands to multiple containe

- **SEND_AND_WAIT** Send all commands regardless of the container state and wait for containers to recover/become available to complete the overall command. Setting this option may result in waiting forever if a container is not reachable or has been disabled. Generally, this should only be used if you are confident of the return of unavailable container or, the user that caused the invocation is happy with the possibility of a long wait.
- **SEND_TO_RUNNING** Send commands only to those containers that are running at the time the command was issued. This can cause incomplete results to be returned (searching only a subset of the intended containers for example) but it is generally better that using SEND_AND_WAIT. It's possible that just after checking container availability a container becomes unavailable and the command still has to wait for an extended period while the container recovers. However, is relatively unlikely.
- **CANCEL_UNLESS_COMPLETE** Only send commands if all the containers required are running. As with SEND_TO_RUNNING it's possible that just after checking container availability a container becomes unavailable and the command has to wait for an extended period while the container recovers. However, is relatively unlikely and the possibility of sending a command to a permanently unavailable container is avoided.

Definition at line 1182 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.SEND_TO_RUNNING = 1 `[static]`

See also:
SEND_AND_WAIT

Definition at line 1184 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.CANCEL_UNLESS_COMPLETE = 2 `[static]`

See also:
SEND_AND_WAIT

Definition at line 1186 of file ISAFClient.java.

final int com.orimos.saf.client.ISAFClient.IMPORT_CREATE_TRANSIENT = 0x10 `[static]`

Flags that control how the import is performed.

- **IMPORT_CREATE_TRANSIENT**SAOs imported are all made transient regardless of their persistent/transient state in the export
- **IMPORT_IGNORE_MISMATCHED_LIBS**If libraries cannot be matched against their name and id the export would normally fail. This flag forces the import to continue with the possible consequence that; incorrect SAOs, SAOs cannot be created or SAOs may be incorrectly linked.

  Use with extreme caution.
- **IMPORT_IGNORE_MISMATCHED_INPUTS**If external inputs cannot be matched in the importing application the import would normally fail. This flag forces the import to continue with the consequence that SAOs will not be linked and the imported hierarchy may not function.
- **IMPORT_OWNER_GROUP**If this flag is set the import will apply permissioning settings in the export file. This option is currently unsupported, and supplying it will have no effect.

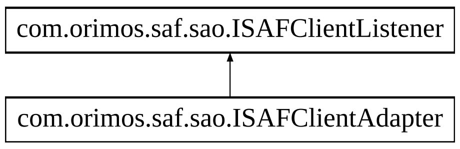
Definition at line 1262 of file ISAFClient.java.

The documentation for this interface was generated from the following file:

- ISAFClient.java

A.3 com.orimos.saf.sao.ISAFClientAdapter Class Reference

Inheritance diagram for com.orimos.saf.sao.ISAFClientAdapter::


com.orimos.saf.sao.ISAFClientListener
com.orimos.saf.sao.ISAFClientAdapter

A.3.1 Detailed Description

This is a adapter class, useful where only some of the methods on the ISAFClientListener interface need to be implemented.

Definition at line 61 of file ISAFClientAdapter.java.

Public Member Functions

- void batchEnd (ISAFClient client)

  Called at the end of a batch of updates/responses.

- void batchStart (ISAFClient client)

  A mechanism exists to allow groups of updates and responses to be grouped together.

- void noLicence (ISAFClient client)

  Called to notify the client that there is not a valid licence available for this product.

- void statusChange (ISAFClient client, SAFClientStatus status)

  Called to notify the client of important changes of state in the SAF.

- void terminated (ISAFClient client)

  Called to acknowledge that the terminated command has been actioned.

- boolean exceptionCaught (Throwable exception)

  An uncaught exception was detected in user code called from the client API.

A.3.2 Member Function Documentation

void com.orimos.saf.sao.ISAFClientAdapter.batchEnd (ISAFClient *client*)

Called at the end of a batch of updates/responses.

Parameters:
  client The client which ”owns” this ISAFClientListener.

Implements com.orimos.saf.sao.ISAFClientListener.

Definition at line 62 of file ISAFClientAdapter.java.

```
62 {}
```

void com.orimos.saf.sao.ISAFClientAdapter.batchStart (ISAFClient *client*)

A mechanism exists to allow groups of updates and responses to be grouped together.

They arrive in batches - this call is made before the first callback of a batch. A call to batchEnd is made afterwards. Note that some callbacks may still be made outside the batch mechanism and that there may not be any callbacks between the batchStart and batchEnd. You should not change any subscriptions inside these callbacks.

Parameters:

client The client which ”owns” this ISAFClientListener.

Implements com.orimos.saf.sao.ISAFClientListener.

Definition at line 63 of file ISAFClientAdapter.java.

```
63 {}
```

void com.orimos.saf.sao.ISAFClientAdapter.noLicence (ISAFClient *client*)

Called to notify the client that there is not a valid licence available for this product.

If there is a problem with the licence then this method will be called back once soon after the client is created (but only if SAFClient.initialise() is called - as the documentation says it should be!). The program will continue to run but will be unable to successfully use SAF's facilities.

Parameters:

client The client which ”owns” this ISAFClientListener.

Implements com.orimos.saf.sao.ISAFClientListener.

Definition at line 64 of file ISAFClientAdapter.java.

```
64 {}
```

void com.orimos.saf.sao.ISAFClientAdapter.statusChange (ISAFClient *client*, SAFClientStatus *status*)

Called to notify the client of important changes of state in the SAF.

The ISAFClientStatus is valid for the duration of the call and should be copied if it is required outside the call.

Parameters:

client The client which ”owns” this ISAFClientListener.

status The status

Implements com.orimos.saf.sao.ISAFClientListener.

Definition at line 65 of file ISAFClientAdapter.java.

```
65 {}
```

void com.orimos.saf.sao.ISAFClientAdapter.terminated (ISAFClient *client*)

Called to acknowledge that the terminated command has been actioned.

This will be the last callback made for this ISAFClient. This means you can let this ISAFClient-Listener go out of scope.

Parameters:

client The client which ”owns” this ISAFClientListener.

Implements com.orimos.saf.sao.ISAFClientListener.

Definition at line 66 of file ISAFClientAdapter.java.

```
66 {}
```

boolean com.orimos.saf.sao.ISAFClientAdapter.exceptionCaught (Throwable *exception*)

An uncaught exception was detected in user code called from the client API.

The *strongly* recommended course of action is to log the error and then to terminate your application. The client API traps these exceptions so that it can report them to the client directly rather than trying to handle them in native code where where the call has originated.

If the client implementation is for a Swing based client you should not display a modal dialog during this call because it will block the client interface and the client may be terminated if it remains blocked for a sufficient length of time.

If the implementation of this method throws an exception it will be caught and logged.

This method will only be called on the ISAFClientListener supplied when the client was created.

Parameters:

exception The exception that was caught by the client API.

Returns:

true if the method handled the exception. If true then the client API will do nothing. If false the client API will log the throwable via its logger.

Implements com.orimos.saf.sao.ISAFClientListener.

Definition at line 67 of file ISAFClientAdapter.java.

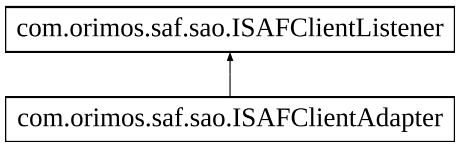
```
67 {return false;}
```

The documentation for this class was generated from the following file:

- ISAFClientAdapter.java

A.4 com.orimos.saf.sao.ISAFClientListener Interface Reference

Inheritance diagram for com.orimos.saf.sao.ISAFClientListener::


com.orimos.saf.sao.ISAFClientListener
com.orimos.saf.sao.ISAFClientAdapter

A.4.1 Detailed Description

This class is used to receive notifications of changes in state of the connection to the SAF and of errors that are not reported via SAOs themselves.

Definition at line 18 of file ISAFClientListener.java.

Public Member Functions

- void noLicence (ISAFClient client)

  Called to notify the client that there is not a valid licence available for this product.

Package Functions

- void statusChange (ISAFClient client, SAFClientStatus status)

  Called to notify the client of important changes of state in the SAF.

- void batchStart (ISAFClient client)

  A mechanism exists to allow groups of updates and responses to be grouped together.

- void batchEnd (ISAFClient client)

  Called at the end of a batch of updates/responses.

- void terminated (ISAFClient client)

  Called to acknowledge that the terminated command has been actioned.

- boolean exceptionCaught (Throwable exception)

  An uncaught exception was detected in user code called from the client API.

A.4.2 Member Function Documentation

void com.orimos.saf.sao.ISAFClientListener.noLicence (ISAFClient *client*)

Called to notify the client that there is not a valid licence available for this product.

If there is a problem with the licence then this method will be called back once soon after the client is created (but only if SAFClient.initialise() is called - as the documentation says it should be!). The program will continue to run but will be unable to successfully use SAF's facilities.

Parameters:

client The client which "owns" this ISAFClientListener.

Implemented in com.orimos.saf.sao.ISAFClientAdapter.

void com.orimos.saf.sao.ISAFClientListener.statusChange (ISAFClient *client*, SAFClientStatus *status*) `[package]`

Called to notify the client of important changes of state in the SAF.

The ISAFClientStatus is valid for the duration of the call and should be copied if it is required outside the call.

Parameters:

client The client which "owns" this ISAFClientListener.

status The status

Implemented in com.orimos.saf.sao.ISAFClientAdapter.

void com.orimos.saf.sao.ISAFClientListener.batchStart (ISAFClient *client*) `[package]`

A mechanism exists to allow groups of updates and responses to be grouped together.

They arrive in batches - this call is made before the first callback of a batch. A call to batchEnd is made afterwards. Note that some callbacks may still be made outside the batch mechanism and that there may not be any callbacks between the batchStart and batchEnd. You should not change any subscriptions inside these callbacks.

Parameters:

client The client which "owns" this ISAFClientListener.

Implemented in com.orimos.saf.sao.ISAFClientAdapter.

void com.orimos.saf.sao.ISAFClientListener.batchEnd (ISAFClient *client*) `[package]`

Called at the end of a batch of updates/responses.

Parameters:

client The client which "owns" this ISAFClientListener.

Implemented in com.orimos.saf.sao.ISAFClientAdapter.

void com.orimos.saf.sao.ISAFClientListener.terminated (ISAFClient *client*) `[package]`

Called to acknowledge that the terminated command has been actioned.

This will be the last callback made for this ISAFClient. This means you can let this ISAFClientListener go out of scope.

Parameters:

client The client which "owns" this ISAFClientListener.

Implemented in com.orimos.saf.sao.ISAFClientAdapter.

boolean com.orimos.saf.sao.ISAFClientListener.exceptionCaught (Throwable *exception*) `[package]`

An uncaught exception was detected in user code called from the client API.

The *strongly* recommended course of action is to log the error and then to terminate your application. The client API traps these exceptions so that it can report them to the client directly rather than trying to handle them in native code where where the call has originated.

If the client implementation is for a Swing based client you should not display a modal dialog during this call because it will block the client interface and the client may be terminated if it remains blocked for a sufficient length of time.

If the implementation of this method throws an exception it will be caught and logged.

This method will only be called on the ISAFClientListener supplied when the client was created.

Parameters:

exception The exception that was caught by the client API.

Returns:

true if the method handled the exception. If true then the client API will do nothing. If false the client API will log the throwable via its logger.

Implemented in com.orimos.saf.sao.ISAFClientAdapter.

The documentation for this interface was generated from the following file:

- ISAFClientListener.java

A.5 com.orimos.saf.sao.ISAOListener Interface Reference

A.5.1 Detailed Description

This is the class that must be implemented to receive notifications of responses to requested SAOs, updates to those SAOs and status changes in SAOs.

Each call returns a reference to an SAO instance accessible by the local client.

The minimum information contained by all SAO instances will be the name, Id, current value and the names and Ids of all of the children. SAOs will normally be accessed during callbacks. If you choose to access SAOs outside of the callbacks then you are responsible for placing a lock on the ISAFClient.

If an SAO which you have on request is deleted then you will get a response() callback with a status of SAF_DELETED on the deleted SAO - you may then cancel the request. In addition, if the parent of the deleted SAO is on request then the parent will receive a childRemoved() callback; the response() callback on the child is guaranteed to happen before the childRemoved() on the parent.

Each request made is terminated by a cancelled(), succeeded() or failed() callback. cancelled() as the result of a cancel command; failed() or succeeded() as the result of commands. If the request has its own ISAOListener instance then you may use this callback to remove the ISAOListener instance.

Note that several requestIds can refer to the same SAO; a change to this SAO will result in a callback on each Id. You cannot guarantee any particular order to callbacks except that you will always get a response() before an update() which changes data. The first callback on an Id could be an update() when the SAO is temporarily or permanently unavailable. You might get several responses callbacks with or without updates between.

Requesting a non-existent SAO will result in a failed callback.

Each of these callbacks takes a reference to a IResult which contains the SAO associated with the request (NULL in the case of a cancelled() or succeeded() callback), the requestId itself and the associated ISAFClient, as well as an optional command result.

Each of the cancelled(), failed() or succeeded() callbacks is an indication that there will be no more callbacks on that request id - this callback indicates that you may release the ISAOListener associated with the request.

Definition at line 53 of file ISAOListener.java.

Public Member Functions

- void response (IResult data)

  Called to notify that a full value for a requested SAO is available.

Package Functions

- void update (IResult data)

  Called to notify that the update value for a requested SAO are available.

- void renamed (IResult data)

  Called to notify that the name of a requested SAO has changed.

- void childAdded (IResult data, ISAO child)

  This method is called when a child has been added to an SAO you have requested.

- void childRemoved (IResult data, SAOId id, String name, ISAO next)

  This method is called when a child has been removed from a parent you have requested with children.

- void childRenamed (IResult data, ISAO child)

  This method is called when the child of an SAO you have requested with children has been renamed.

- void cancelled (IResult data)

  Called to acknowledge that an SAO cancel has been actioned and that no furthur callbacks will be made on the requestId.

- void failed (IResult data)

  Called to inform that a command or request has failed and that no furthur callbacks will be made on the requestId.

- void succeeded (IResult data)

  Called to acknowledge that an operation has been actioned, has succeeded and that no furthur callbacks will be made on the requestId.

A.5.2 Member Function Documentation

void com.orimos.saf.sao.ISAOListener.response (IResult *data*)

Called to notify that a full value for a requested SAO is available.

If the request was snapshot then this callback will be shortly followed by a cancelled callback.

Parameters:
  data Details of the response.

void com.orimos.saf.sao.ISAOListener.update (IResult *data*) `[package]`

Called to notify that the update value for a requested SAO are available.

Parameters:
  data Details of the response.

void com.orimos.saf.sao.ISAOListener.renamed (IResult *data*) `[package]`

Called to notify that the name of a requested SAO has changed.

Parameters:

data Details of the response.

void com.orimos.saf.sao.ISAOListener.childAdded (IResult *data*, ISAO *child*) `[package]`

This method is called when a child has been added to an SAO you have requested.

Parameters:

data Details of the response. The ISAO is that of the parent of the added child.

child The child that has been added.

void com.orimos.saf.sao.ISAOListener.childRemoved (IResult *data*, SAOId *id*, String *name*, ISAO *next*) `[package]`

This method is called when a child has been removed from a parent you have requested with children.

Removal may be either because the child has been deleted or because it has been moved in the hierarchy.

Parameters:

data Details of the response. The ISAO is that of the parent of the removed child.

id The Id of the removed child.

name The name of the removed child.

next If not NULL then this SAO is the one following the removed one in the parent; if NULL then the removed one was the last child.

void com.orimos.saf.sao.ISAOListener.childRenamed (IResult *data*, ISAO *child*) `[package]`

This method is called when the child of an SAO you have requested with children has been renamed.

Parameters:

data Details of the response. The ISAO is that of the parent of the renamed child.

child The child, which already has its new name.

void com.orimos.saf.sao.ISAOListener.cancelled (IResult *data*) `[package]`

Called to acknowledge that an SAO cancel has been actioned and that no furthur callbacks will be made on the requestId.

Will be automatically generated after a snapshot request. The result string might be an error message.

Parameters:

data Details of the response. The ISAO is null.

void com.orimos.saf.sao.ISAOListener.failed (IResult *data*) `[package]`

Called to inform that a command or request has failed and that no furthur callbacks will be made on the requestId.

The result string might be an error message.

Parameters:

data Details of the response. The ISAO is null.

void com.orimos.saf.sao.ISAOListener.succeeded (IResult *data*) `[package]`

Called to acknowledge that an operation has been actioned, has succeeded and that no furthur callbacks will be made on the requestId.

The operation may return a result in commandResult.

Parameters:

data Details of the response. The ISAO is null.

The documentation for this interface was generated from the following file:

- ISAOListener.java

A.6 com.orimos.saf.sao.ISAOSpec Interface Reference

A.6.1 Detailed Description

This interface is used when specifying the identity of an SAO.

An SAO can be specified either by its SAOId, by a series of SAO names from the _root through the hierarchy or by a string specifying the same information. In the case of either of these last two an SAO may be specified relative to an existing SAO, as long as the full path to that SAO is available.

To complete an SAO specification you should use assign, which will clear down any previous contents, followed by adding any further components required. If the ISAOSpec is badly formed when it is used (primarily when making a request on the CLientAPI) then the request will be rejected. Having no components constitutes means it is not well formed.

Note that when passing a string containing a path to an SAO there must be some way of delimiting the components. The default is to separate the components with a '/' character and allow any '/' or '\' character to be escaped with a '\' character.

Definition at line 88 of file ISAOSpec.java.

Public Member Functions

- ISAOSpec assign (SAOId id)

  Specify using a SAOId.

Package Functions

- ISAOSpec assign (String component, ISAO base)

  Specify the first component.

- ISAOSpec assign (ISAO base, String name)

  Specify as a parsed string.

- ISAOSpec assign (ISAFValue value)

  Specify as an ISAFValue.

- ISAOSpec assign (ISAOSpec value)

  Specify as an ISAOSpec.

- ISAOSpec add (String component)

  Add a component to the specification.

- ISAOSpec addPath (String name)

  Add a parsed string to the specification.

- int getCId ()

  Will return NOCONTAINER if !isOK else returns the target id.

- boolean isById ()

  Returns true if this is specified by id.

- SAOId getId ()

  Returns a reference to the Id.

- ISAFValue makeValue ()

  Ownership of the new object is given to the caller.

- boolean isEqual (ISAOSpec other)

  Is this ISAOSpec equal to the ”other” one.

- boolean append (ISAFValue value)

  Append from an ISAFValue.

- Object **clone** () throws CloneNotSupportedException

A.6.2 Member Function Documentation

ISAOSpec com.orimos.saf.sao.ISAOSpec.assign (SAOId *id*)

Specify using a SAOId.

Returns:
A reference to itself.

ISAOSpec com.orimos.saf.sao.ISAOSpec.assign (String *component*, ISAO *base*) `[package]`

Specify the first component.

Parameters:

component The first component when building the specification a component at a time. No parsing of the name occurs and there are no escape conventions. If NULL then no components are present.

base If a relative path the place from where the path relates. If NULL then this is not a relative path. The ISAO is used immediately to extract any required information so can be cancelled after this call.

Returns:
A reference to itself.

ISAOSpec com.orimos.saf.sao.ISAOSpec.assign (ISAO *base*, String *name*) `[package]`

Specify as a parsed string.

Note that the parsed string is immediately turned into a component list so you may add components afterwards.

Parameters:

base If a relative path the place from where the path relates. If NULL then this is not a relative path. The ISAO is used immediately to extract any required information so can be cancelled after this call.

name The path to the SAO. Parsed into components. Leading and trailing delimiter characters will be ignored. both character can be specified in a component name. These characters should be different.

Returns:

A reference to itself.

ISAOSpec com.orimos.saf.sao.ISAOSpec.assign (ISAFValue *value*) `[package]`

Specify as an ISAFValue.

It is assumed that value was produced by makeValue().

Parameters:

value The value containing the specification.

Returns:

A reference to itself.

ISAOSpec com.orimos.saf.sao.ISAOSpec.assign (ISAOSpec *value*) `[package]`

Specify as an ISAOSpec.

Parameters:

value The value containing the specification.

Returns:

A reference to itself.

ISAOSpec com.orimos.saf.sao.ISAOSpec.add (String *component*) `[package]`

Add a component to the specification.

If the last assign() was for id by SAOId, or there has been no assign() then ignored.

Parameters:

component The next piece. No parsing of the name occurs and there are no escape conventions. If NULL then ignored.

Returns:

A reference to itself.

ISAOSpec com.orimos.saf.sao.ISAOSpec.addPath (String *name*) `[package]`

Add a parsed string to the specification.

Note that the parsed string is immediately turned into a component list so you may add components afterwards. If the last assign() was for id by SAOId, or there has been no assign() then ignored.

Parameters:

name The path to the SAO. Parsed into components. Leading and trailing delimiter characters will be ignored.

Returns:

A reference to itself.

int com.orimos.saf.sao.ISAOSpec.getCId () `[package]`

Will return NOCONTAINER if !isOK else returns the target id.

SAOId com.orimos.saf.sao.ISAOSpec.getId () `[package]`

Returns a reference to the Id.

ISAFValue com.orimos.saf.sao.ISAOSpec.makeValue () `[package]`

Ownership of the new object is given to the caller.

The object returned from the getValue() call can be serialized to retain ISAOSpec values between application invocations.

Returns:

An ISAFValue which represents the same value as in this object

boolean com.orimos.saf.sao.ISAOSpec.isEqual (ISAOSpec *other*) `[package]`

Is this ISAOSpec equal to the ”other” one.

Two ISAOSpecs are equal if they are both ”by Id” or both not ”by Id” and the ids or paths (as indicated by isById()) are equal.

Parameters:

other The ISAOSpec to compare against.

Returns:

true if equal otherwise false.

boolean com.orimos.saf.sao.ISAOSpec.append (ISAFValue *value*) `[package]`

Append from an ISAFValue.

It is assumed that value was produced by makeValue(). Returns false if the two components cannot be merged, possibly because one of them isById().

Parameters:

value The value containing the specification.

Returns:

True if the join succeeded.

The documentation for this interface was generated from the following file:

- ISAOSpec.java

A.7 JClientSample.FieldRequest Class Reference

A.7.1 Detailed Description

The class represents a request to a currency SAO, recording the currency and the SAO name.

Definition at line 168 of file JClientSample.java.

Public Member Functions

- FieldRequest (String currency, String field)

  Constructor.

- String getCurrency ()

  Get the currency.

- String getField ()

  Get the field.

Package Attributes

- String _currency

  Which currency this request is for.

- String _field

  Which field this request is for.

A.7.2 Member Function Documentation

String JClientSample.FieldRequest.getCurrency ()

Get the currency.

Definition at line 187 of file JClientSample.java.

```
187                             {
188             return _currency;
189         }
```

String JClientSample.FieldRequest.getField ()

Get the field.

Definition at line 194 of file JClientSample.java.

```
194                           {
195             return _field;
196         }
```

The documentation for this class was generated from the following file:

- JClientSample.java

A.8 JClientSample.MyClientListener Class Reference

A.8.1 Detailed Description

Implements the SAF Client level callbacks.

The class leaves as stubs, methods in which we have no interest

Definition at line 200 of file JClientSample.java.

Public Member Functions

- void **noLicence** (ISAFClient client)
- void **statusChange** (ISAFClient client, SAFClientStatus status)
- void **batchStart** (ISAFClient client)
- void **batchEnd** (ISAFClient client)

The documentation for this class was generated from the following file:

- JClientSample.java

A.9 JClientSample.MySAOListener Class Reference

A.9.1 Detailed Description

An implementation of the ISAOListener interface.

Leaves as stubs, any methods that are not required for this sample.

Definition at line 231 of file JClientSample.java.

Public Member Functions

- void **response** (IResult data)
- void **update** (IResult data)
- void **cancelled** (IResult data)
- void **failed** (IResult data)

The documentation for this class was generated from the following file:

- JClientSample.java

A.10 JSwingClientSample.MyRenderer Class Reference

A.10.1 Detailed Description

This Renderer class controls the formatting of doubles, so we get 3 decimal places and right alignment.

Definition at line 202 of file JSwingClientSample.java.

Public Member Functions

- Component **getTableCellRendererComponent** (JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column)

Package Attributes

- DecimalFormat _**decimalFormat** = new DecimalFormat("#0.000")

The documentation for this class was generated from the following file:

- JSwingClientSample.java

A.11 JSwingClientSample.MyTableModel Class Reference

A.11.1 Detailed Description

This table model class obtains and holds the data from SAF.

Definition at line 228 of file JSwingClientSample.java.

Public Member Functions

- int **getColumnCount** ()
- int **getRowCount** ()
- Object **getValueAt** (int row, int column)
- String **getColumnName** (int column)

Package Functions

- **MyTableModel** (ISAFClient client, String[ ] currencies)

Package Attributes

- String[ ] _colHeadings = {"Currency", "Bid", "Ask", "Mid", "XRate"}

  The column heading.

The documentation for this class was generated from the following file:

- JSwingClientSample.java

A.12 JSwingClientSample.MyTableModel.FieldRequest Class Reference

A.12.1 Detailed Description

FieldRequest maps a request for a single SAO to the grid element that displays it.

Definition at line 412 of file JSwingClientSample.java.

Public Member Functions

- FieldRequest (int row, int col)

  Constructor.

- int getRow ()

  Get the row for this request.

- int getCol ()

  Get the column for this request.

- void setSAO (ISAO sao)

  Set the SAO for this request.

- ISAO getSAO ()

  Return the SAO for this request.

A.12.2 Constructor & Destructor Documentation

JSwingClientSample.MyTableModel.FieldRequest.FieldRequest (int *row*, int *col*)

Constructor.

Parameters:

row The row where this field is to be displayed.

col The column where this field is to be displayed.

Definition at line 428 of file JSwingClientSample.java.

```
428                                                   {
429                 _row = row;
430                 _col = col;
431             }
```

A.12.3 Member Function Documentation

int JSwingClientSample.MyTableModel.FieldRequest.getRow ()

Get the row for this request.

Returns:
The row for this request.

Definition at line 437 of file JSwingClientSample.java.

```
437                                  {
438                 return _row;
439             }
```

int JSwingClientSample.MyTableModel.FieldRequest.getCol ()

Get the column for this request.

Returns:
The column for this request.

Definition at line 445 of file JSwingClientSample.java.

```
445                                  {
446                 return _col;
447             }
```

void JSwingClientSample.MyTableModel.FieldRequest.setSAO (ISAO *sao*)

Set the SAO for this request.

Definition at line 452 of file JSwingClientSample.java.

```
452                                           {
453                 _sao = sao;
454             }
```

ISAO JSwingClientSample.MyTableModel.FieldRequest.getSAO ()

Return the SAO for this request.

Definition at line 459 of file JSwingClientSample.java.

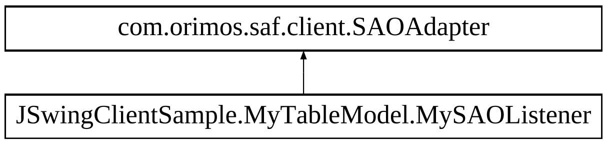
```
459                                   {
460                 return _sao;
461             }
```

The documentation for this class was generated from the following file:

- JSwingClientSample.java

A.13 com.orimos.saf.client.SAOAdapter Class Reference

Inheritance diagram for com.orimos.saf.client.SAOAdapter::


com.orimos.saf.client.SAOAdapter
JSwingClientSample.MyTableModel.MySAOListener

A.13.1 Detailed Description

A Simple adapter class that provides default implementations of all the SAOListener class methods.

Definition at line 71 of file SAOAdapter.java.

Public Member Functions

- void **cancelled** (IResult data)
- void **failed** (IResult data)
- void **childAdded** (IResult data, ISAO child)
- void **childRemoved** (IResult data, SAOId id, String name, ISAO next)
- void **childRenamed** (IResult data, ISAO child)
- void **response** (IResult data)
- void **succeeded** (IResult data)
- void **update** (IResult data)
- void **renamed** (IResult data)

The documentation for this class was generated from the following file:

- SAOAdapter.java

Appendix B Compiling Java Programs

The creation of **SONARIS/Framework** client programs in java is achieved using the classes and interfaces in the `safjclient.jar` file which must be specified in the java compiler's classpath.

Windows

To compile the example programs supplied with the **Software Development Kit** from the command line, change to the `SDK\examples\JavaClient` directory under the **SONARIS/Framework** installation directory and execute the following commands:

```
javac -g -classpath ..\..\..\safjclient.jar JClientSample.java
javac -g -classpath ..\..\..\safjclient.jar JSwingClientSample.java
```

To run the examples, execute the following commands (note that the classes supplied by the `trove.jar` file are required):

```
java -classpath ..\..\..\safjclient.jar;..\..\..\trove.jar;\. JClientSample
java -classpath ..\..\..\safjclient.jar;..\..\..\trove.jar;\. JSwingClientSample
```

Glossary

Active Interface
An interface to a *SONARIS/Framework* programming object for which a standard implementation is available. Instances of objects which provide this implementation are supplied by the *SONARIS/Framework Runtime Environment* and can be queried and manipulated via their interface definition. See also *Listener Interface*.

Application
In *SONARIS/Framework* terms, a complete solution composed of an *application service* which processes and publishes data using *SAOs*, and *client* programs capable of accessing the data held by those SAOs.

Application Service
In *SONARIS/Framework* terms, a group of *containers* hosted by *processes* using *SAOs* to implement a service capable of supplying the data held by those SAOs to *client* programs.

Child
In terms of *SAO hierarchies*, an *SAO* which is parented by another SAO within a hierarchy is said to be a *child* of that SAO.

Client
In *SONARIS/Framework* terms, a program capable of subscribing to the data held by *SAOs* which form part of an *application service*.

Configuration Database
A database (usually hosted by some RDBMS system) which acts as a store for configuration information that describes *hierarchies* of *SAOs*, the properties of those SAOs (value, history, log level etc.), the *connections* between them, and by which *container* they are hosted as part of an *application service* (refer to the *SONARIS/Framework - Release Notes* for details of the RDBMS systems supported).

Connection
A *dependency* relationship between two *SAOs* which specifies that the value of one of the SAOs is dependant on the value of the other, and that changes in the value of the former will be reflected in the value of the later, or that the later will take some action based on the change of the former.

Container
An object created by a `saf` *process* responsible for hosting a *hierarchy* of *SAOs* and servicing them with data supplied from other containers, *clients* or *External System Retrievers*. A container uses a single execution thread to deliver *updates* to SAOs and then, by means of a *ripple*, propagate resultant changes to other *dependant* SAOs and supply updates to other containers or clients which hold subscriptions to the affected SAOs.

Cycle
A pattern in the *dependency graph* of *connections* between *SAOs* which revisits an SAO during a single *ripple*. The most direct form of a cycle occurs when an SAO uses its own *output* value as one of its own inputs. Cycles in the dependency graph can be used to give controlled *feedback* effects, but should be used with care, as they can yield results which are difficult to interpret.

Data Source
In *SONARIS/Framework* terms, a source of configuration data which can be read in order to initialise a SONARIS/Framework *application service* (usually a relational database). See *Configuration Database*.

Dependant
In *SONARIS/Framework* terms, an *SAO* whose value is established and maintained based on a *connection* which uses the *output* value of another SAO as one of its *inputs* is said to be dependant on that other SAO.

Dependency Graph
The set of *dependency* relationships established as a result of forming *connections* between *SAOs*.

ESR
See *External System Retriever*.

ESS
See *External System Sender*.

Export
The process of creating an XML description of a *hierarchy* of *SAOs* which can then be used to create a copy of that hierarchy in another location in, possibly, another *application service*. See also *Import File*.

External System Retriever
An implementation of the **IExternalSystemRetriever** interface (defined in `isafesr.h`) which wraps third party software provided by an external information supplier allowing data to be passed to individual SAOs within a container. An implementation of an ESR will typically map the information available from the external supplier into a hierarchical namespace which corresponds to hierarchies of SAOs that can be created within application containers. If the supplier being wrapped also supports the return of data for external publishing or contribution a combined IExternalSystemSender interface may also be supplied.

External System Sender
An implementation of the **IExternalSystemSender** interface (defined in `isafess.h`) which wraps third party software provided by an external information publisher allowing data to be sent from individual SAOs within a container for publication by the external system. Like an *ESR*, an implementation of an ESS will typically map the third party structures via which data can be published into a hierarchical namespace which corresponds to hierarchies of SAOs that can be created within application containers.

Feedback
A condition arising from a *cycle* in the *dependency graph* where the value of an *SAO* is set during a *ripple* but not used by one of its *dependants* until a later ripple.

Hierarchy
A complete tree, or branch of a greater tree, of *SAOs* hosted by one or more *containers*. The complete hierarchy representing an *application* is usually composed of several individual hierarchies hosted by separate containers distributed across numerous *machines* and *processes* and joined together at *mount points*.

Import File
An XML file containing a description of a *hierarchy* of *SAOs*. Generally, such files also contain information about *connections* between SAOs in the file as well as their values. Import files can be used to create complex hierarchies of connected SAOs from a single command.

Input
Part of the *definition* of an *SAO* which specifies the end point of a *connection* formed from some other SAO to this in order that this SAO be able to use the *output* value of that SAO in establishing and maintaining its own value.

LCS
See *Licence/Configuration Server*.

Licence/Configuration Server
A *node* within a *partition* configured to manage configuration, licencing and connection topology for the other nodes within the partition.

Listener Interface
An interface to a *SONARIS/Framework* programming object which must be implemented by a program or component wishing to communicate with *SONARIS/Framework Runtime Environment*. See also *Active Interface*.

Machine
In *SONARIS/Framework* terms, a host on which *processes* hosting *containers* are run to provide all or part of an *application service.*

Mount Point
A defined location within a *hierarchy* of *SAOs* hosted by a *container* at which another hierarchy, hosted by another container is rooted.

Node
A machine on which *SONARIS/Messaging* is installed and which forms messaging connections to other nodes within the same *partition*.

Output
The current value of an *SAO* available to any *client* which holds a subscription to it, or to any other SAO which uses it as one of its *inputs*.

Parent In terms of *SONARIS/Framework hierarchies*, an *SAO* which has one or more *child* SAOs is said to act as the *parent* for those SAOs.

Partition
A group of *nodes* one of which is configured to act as *LCS* for the others.

Persistent SAO
An instance of an *SAO* whose details (including Id, name, *hierarchical* location, ownership details, *input* details etc.) and (optionally) value, are stored in the *configuration database* for the *application* such that the SAO will be available every time the *container* which hosts it is run.

Process
In *SONARIS/Framework* terms, an individual invocation of the `saf` program which creates and manages one or more *containers* to provide part or all of an *application* service.

Proxy SAO
A subset of the full functionality of an *SAO* used to remotely represent an SAO in either a *client* program or a *container* other than the one which hosts it.

Ripple
A sequence of calls by a *container* to methods of a subset of the *SAOs* managed by that container to ensure that the values held by those SAOs is brought up to date with respect to one another, and to pass on any value changes which have occurred to *client* programs which hold a subscription to those SAOs. A ripple is usually performed as a result of new data being supplied as *updates* to one or more SAOs from some external source (another container, a client, an *ESR* etc.) to allow any *dependant* SAOs to modify their values accordingly.

SAO
See *SONARIS Application Object*.

SAO Definition
A formal description of an *SAO* that provides a general description of the SAO, defines the type of data it is capable of storing, and specifies the number, types, usage conditions, and general descriptions of its *inputs*. Such definitions are supplied with every *SAO library* as XML text, and must be loaded by every *application* which makes use of that library to verify appropriate usage of instances of the SAOs within the application.

SAO Library
A library of *SAOs* supplied in two parts; a dynamic linked library binary file specific to a particular computer architecture, and an architecture and operating system neutral *SAO library definition* file which formally describes the SAOs provided by the library using XML.

Sibling
In terms of *SONARIS/Framework hierarchies*, an *SAO* whose *parent* also acts as a parent for one or more other SAOs is said to be a *sibling* of those SAOs.

SONARIS/Framework Runtime Environment The core features and services of the *SONARIS/Framework* system available to client programs and SAOs.

SONARIS/Framework
Component based, distributed development framework for implementing real-time calculation and data processing solutions.

SONARIS Application Object
The most basic component of a *SONARIS/Framework application*. SAO instances are hosted by *containers* and arranged in *hierarchies*. The implementations of SAOs are supplied by *SAO libraries*.

SONARIS/Messaging
The underlying messaging system used by *SONARIS/Framework* for the exchange of data between SONARIS/Framework *containers* and *clients*.

Template

A branch of a *hierarchy* of *SAOs* which serves as a master copy to be replicated at other points in the hierarchy.

Transient SAO

An instance of an *SAO* which, unlike a *persistent SAO*, is only available for the life time of the of the *container* which hosts it. The Id assigned to a transient SAO will not be used again by any other SAO.

Update

A delivery of a new value to an *SAO* (usually from outside of the current *container*) or from a container to a *client*.