Novell ICEcore

1.0

www.novell.com

WEB SERVICES GUIDE

July 14, 2008

Novell®

Legal Notices

Novell, Inc., makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc., makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. See the Novell International Trade Services Web page (http://www.novell.com/info/exports/) for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Novell, Inc., has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed on the Novell Legal Patents Web page (http://www.novell.com/company/legal/patents/) and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the latest online documentation for this and other Novell products, see the Novell Documentation Web page (http://www.novell.com/documentation/team_plus_conf/).

Novell Trademarks: For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials: All third-party trademarks are the property of their respective owners.

Images of personnel in the screen shots supplied by Copyright © Comtech Enterprises, Inc. (http://www.comteche.com)

ICEcore*

Contents

About This Manual

This manual provides reference information for programmers who want to access ICEcore from another program (as opposed to the user interface presented in a web browser). To accomplish this task, programmers use the ICEcore Web services.

This guide includes these topics:

Audience

This guide is intended for programmers.

Software and Documentation Version

This manual describes features in ICEcore Version 1.0.3. This is Version 1.0.1 of this manual.

Additional Documentation

You can find more information in the ICEcore documentation, which is accessible from links within the ICEcore software:

- ICEcore Help Mode
- *ICEcore Quick Start Guide*
- *ICEcore User Guide*
- *ICEcore Installation and Configuration Guide*
- *ICEcore Administration Guide*

Conventions

This manual uses the conventions described in the following paragraphs and table.

A greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a SiteScape trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux or UNIX, should use forward slashes as required by your software.

What you see	What it means
Click the *Add a team workspace* button. Click the *Getting Started* link.	Items that you can click on the page, labeled user interface items, variable or replaceable text (including programming variables and syntax parameters) are presented in *italic* font.
Blog summary - Provides a.... **Note:** Remember that....	Many types of headers and defined terms in a list are presented in bold font.
Type `status`, then press Enter. Open the `ManagerGuide.pdf` file. Use the `open_db` routine with its `lock` parameter.	Filenames, text that you must type, commands, command options, routines, Web service messages, and parameters are presented in `Courier` font when occurring in a body of text.
[page]	Optional syntax parameters are enclosed in brackets ([ ]).
..., paramSyntax1 | paramSyntax2,...	Required parameters accepting two or more optional syntax items are separated by the vertical-line character.
(V1—V1.0.3)	The versions of ICEcore that supports the Web services operation (“all versions between Version 1.0 and Version 1.0.3”).

Web Services Overview

1

Novell® offers a set of messages that you can use in client programs to exchange information with the server (a running installation of ICEcore). This topic includes these sections:

NOTE: This topic documents the Web services supported for ICEcore Version 1.0.3. As of the next major release of this product, all of these messages will be deprecated in favor of new messages; these older messages will continue to function and will be maintained.

1.1 ICEcore Web Services Implementation

ICEcore implements Java Web services, which provide a set of messages and parameters that client programs can use to exchange information with ICEcore. The alphabetized reference section in this documentation provides syntax and examples (Chapter 2, “Web Services Message Reference,” on page 25). In addition, you can access the ICEcore Web Services Description Language (WSDL) file here:

```
http://localhost:8080/ssf/ws/Facade?wsdl
```

Replace the `localhost` specification with the host and port for your ICEcore installation.

NOTE: ICEcore does not currently publish its WSDL file using Universal Description, Discovery, and Integration (UDDI) or the Web Services Inspection Language (WSIL). Use the alphabetized reference section in this manual (Chapter 2, “Web Services Message Reference,” on page 25) or the URL-generated WSDL file to understand the ICEcore message interface.

For ICEcore Version 1.0, regarding messages shown in the product source code, the `search` message is under development and subject to change or deletion at any time. Do not use this message in your client applications.

To implement lower-level details of its Web service calls, ICEcore uses methods included in the Apache Axis toolkit. As a very brief summary, ICEcore code calls Apache Axis methods. These methods convert the message to Simple Object Access Protocol (SOAP) calls. SOAP calls use Remote Procedure Calls (RPC) to call ICEcore Java methods. Conversion of information to and from EXtensible Markup Language (XML) is a part of this process.

NOTE: When presenting Web service examples in this manual, the code includes Apache Axis methods. If you are not using the Apache Axis toolkit, map the Axis methods to the ones provided by your vendor’s Web services toolkit.

Novell highly recommends using the supporting code for the `wsclient.bat` sample as a template for constructing your own Web service calls. This sample client is described in the next section (Section 1.1.1, “Sample Clients,” on page 8).

When your application sends a message and its parameters, ICEcore returns either a string of XML data or a number. Messages that return a number are often ones that create a new object in ICEcore (for example, a folder or an entry), and the number uniquely identifies the newly created object.

The XML returned by ICEcore is free form and does not have a schema. When using ICEcore Web services, you need to read and intuit the XML elements in order to code your application so that it can parse the returned XML.

A subsequent section in this topic provides tips for obtaining values needed for various message parameters (Section 1.5, “Notes about Specific Messages,” on page 13).

This section contains the following subsections:

1.1.1 Sample Clients

ICEcore provides sample clients that can assist you in learning how to use its Web services. These sample clients are located in the ICEcore code base. Visit the Open Source Community page (http://www.icecoreopen.org) for more information about downloading ICEcore source code.

The sample client is located here within the source code:

`/ssf/samples/remotingclient`

Here is a list of sample clients and the first ICEcore version that includes them:

- **`wsclient.bat` (V1+):** This Windows batch file allows you to specify commands and parameters that use the ICEcore Web services. The initial pages of the messages-reference chapter provides a table mapping each of the `wsclient.bat` commands to its corresponding Web services message (Chapter 2, “Web Services Message Reference,” on page 25). Regardless of platform, Novell recommends that you use the Java code that implements this batch file as a template for making ICEcore Web service calls.
- **`wsExport.bat` and `wsImport.bat` (V1.1+):** These Windows batch files take data from a portion of the workspace and folder hierarchy and reproduce it on another file system. These tools are not a complete import and export facility, because they do not retain the workflow states, access-control settings, and history of the original objects.

The following Java file implements the `wsclient.bat` batch file:

`/ssf/samples/remotingclient/src/com/sitescape/team/samples/remoting/client/ws/axis/WSClient.java`

The following file provides sign-in information for `wsclient.bat` (start with the same path as shown in the previous example):

`.../ws/security/PWCallbackText.java`

This is another file used by `wsclient.bat` to log in:

`/ssf/samples/remotingclient/client_deply.wsdd`

Enabling wsclient.bat (Windows systems only)

Before using `wsclient.bat`, you need to do some work in your build to enable it. Using the Eclipse* build environment, click and drag this file into your Ant window:

`/ssf/samples/remotingclient/build.xml`

In the Ant window, open the *wsclient* build file, and double click *zip*.

ICEcore adds the `remotingclient-sample.zip` file to the `/ssf/samples` directory.

To use `wsclient.bat`, do the following:

1. Use a command line window to `cd` to the `/ssf/samples/remotingclient` directory.
2. Type `wsclient.bat`.
3. On the same line, type a `wsclient.bat` command name and desired arguments.

   Command names are listed at the beginning of the topic containing the alphabetized Web services messages (Chapter 2, "Web Services Message Reference," on page 25).
4. Press the Return key.

If the command executes successfully, ICEcore displays the return value in the command line window.

For example, to see an XML string containing summary data for all workspaces and folders that have defined teams, execute the following command in the `/ssf/samples/remotingclient` directory:

```
> wsclient.bat printTeams
```

If you want to execute the `wsclient.bat` file outside of the scope of the build environment's file system, you can unzip the `/ssf/samples/remotingclient-sample.zip` file elsewhere on a Windows file system, use a command line window to `cd` to the location of that copy of the `wsclient.bat` file, and use the batch file there.

1.1.2 ICEcore-Specific Terminology

The following are ICEcore-specific definitions that can assist you when using the ICEcore Web services:

- **binder:** A place, such as a workspace or folder.
- **binder configuration ID:** A number that identifies the template used to create and configure a new workplace or folder. An upcoming section provides additional information about this identifier (Section 1.5.1, "Adding Folders and the Binder Configuration Identifier," on page 13).
- **binder ID:** A unique number that identifies a specific workspace or folder.
- **data item name:** A hidden-tag value that maps an HTML form element with a value stored in the ICEcore database.
- **definition ID:** A unique, 32-character, hexadecimal identifier that maps to a definition for a specific type of entry. (You modify and create definitions using the designers in the administration portlet.) You need to specify this value when creating a new entry in a folder.
- **page:** A hierarchical level in the workspace hierarchy that represents a subset of binders, most often used to group personal workspaces into sets that are convenient for display in the user

interface (UI). An upcoming section provides additional information about this hierarchical level (Section 1.5.7, “Binder Pages and getWorkspaceTreeAsXML,” on page 18).

- **principal:** A registered user or a group.
- **principal ID:** A unique number that identifies a specific user or group.

1.2 Connecting to the Server

To show how a client application connects to the server, this section uses the source code for the `wsclient.bat` batch file to illustrate concepts. Also, the Java source code that enables `wsclient.bat` makes calls to the Apache Axis toolkit methods; if you are not using Apache Axis, map these methods to ones used by your Web services methods.

The `wsclient.bat` batch file is implemented using the methods located in the `WSClient.java` file, as mentioned in the section describing sample clients (Section 1.1.1, “Sample Clients,” on page 8). This file contains the `fetch` method, which is a wrapper that performs most of the direct calls to the Apache Axis toolkit methods, which in turn make the call to Web services. In summary, the `fetch` method connects the `wsclient.bat` client to the server, sets up the remote procedure calls, names the operation, and invokes the passing of messages using Web services.

This section describes step-by-step the initial lines of code in the `fetch` method to show how to connect your client to the server. Consider these initial lines of code:

```
// Replace the hostname in the endpoint appropriately.
String endpoint = "http://localhost:8080/ssf/ws/Facade";
```

The `wsclient.bat file` is designed to be executed on the same machine that runs the ICEcore installation. Many client applications connect to the server from another machine on the Internet. Provide the appropriate host and port for the ICEcore server to which you want to connect.

Consider the next lines of code:

```
// Make sure that...client_deploy.wsdd...is accessible to the program.
EngineConfiguration config = new FileProvider("client_deploy.wsdd");
    .
    .
    .
call.setProperty(WSHandlerConstants.USER, "admin");
```

As required by the source code for the client, the `client_deploy.wsdd` file is located in the same directory as the `wsclient.bat` file. It contains XML needed by Apache Axis to implement security. It establishes a username that a client can use as its persona when working with Web services (by default, it is the `admin` account). It also points to two other files used by Apache Axis as part of the way it implements security:

- `PWCallbackDigest.java`
- `PWCallbackText.java`

A subsection that follows provides helpful notes about security choices for your client application (Section 1.2.1, “Notes About Security,” on page 12).

For example, the `wsclient.bat` client uses the persona of the `admin` ICEcore account. To specify the password for this persona, the `PWCallbackText.java` file contains these lines, which set the password to `test`:

```
if ("admin".equals(id)) {
    pc.setPassword("test");
}
```

The call to the `call.setProperty` method establishes a constant that specifies the `admin` account as being the persona to be used by the client. It is rare that a client uses the same persona for all its work on the server.

Remember that this way of establishing a persona for your client software is specific to Apache Axis. If you are not using Apache Axis, use these examples to inform the work with your Web services toolkit.

These lines of code are the remaining ones required by Apache Axis to set up and then make the call to the server:

```
Service service = new Service(config);

Call call = (Call) service.createCall();

call.setTargetEndpointAddress(new URL(endpoint));

// We are going to invoke the remote operation to fetch the workspace
//  or folder to print.
call.setOperationName(new QName(operation));

// Programmatically set the username. Alternatively you can specify
// the username in the WS deployment descriptor client_deploy.wsdd
// if the username is known at deployment time and does not change
// between calls, which is rarely the case in Aspen.
call.setProperty(WSHandlerConstants.USER, "admin");

if(filename != null) {
    DataHandler dhSource = new DataHandler(new FileDataSource(new
File(filename)));

    call.addAttachmentPart(dhSource); //Add the file.

    call.setProperty(Call.ATTACHMENT_ENCAPSULATION_FORMAT,
Call.ATTACHMENT_ENCAPSULATION_FORMAT_DIME);
} //End of if

Object result = call.invoke(args);
    .
    .
    .
```

1.2.1 Notes About Security

Your client application can implement security in several ways. Each method has advantages and disadvantages, which you should consider. This section contains the following subsections, in order from the most secure to the least secure:

- "Password Digest" on page 12
- "Password Text" on page 12

Password Digest

On the client side of the Web services transaction, the client code provides a username and password, and the Web services security framework digests the password and sends it to the server.

On the server side, ICEcore retrieves the user's password from its database and passes it to the Web services security framework.

One of the disadvantages to this method of implementing security is that, because ICEcore stores the password in encrypted form, it cannot pass a clear-text password to the framework for comparison. To solve this problem, before digest and transmission, the client can use the ICEcore password-encryptor class to apply the same encryptor to the clear-text password and can then pass the encrypted password to the security framework.

So, although secure, this method requires work on the client side.

Password Text

On the client side of the Web services transaction, the client code provides a username and password to the Web services security framework, and the framework passes the password as plain text.

On the server side, the security framework allows ICEcore to retrieve the clear-text password from the message using an application programming interface (API) call. When retrieved, ICEcore applies its internal password encryptor and compares the result with the password stored in the database for the user. ICEcore performs the authentication; if successful, ICEcore returns the clear-text password to the Web services security framework to complete its process.

The primary disadvantage of this method is that it is not secure unless it's being done using SSL. Another disadvantage is that, if the password does not match, ICEcore must throw an exception in order to cause the security framework to abort its process. (This problem with mismatching passwords does not occur using Password Digest.)

So, this method is easier to code on the client side, but it has several disadvantages.

1.3 Sending Messages

In the code example at the end of the previous section, these are the two lines that are responsible for sending the message to the server:

```
call.setOperationName(new QName(operation));
    .
    .
    .
Object result = call.invoke(args);
```

The `operation` parameter maps to the names of the messages documented in the alphabetical reference in this manual (Chapter 2, "Web Services Message Reference," on page 25), `addFolderEntry`, `uploadCalendarEntries`, and so on. The `args` parameter maps to the message parameters documented in the reference section.

Using Apache Axis, hard coded calls that send messages from the client to the server appear as follows:

```
call.setOperationName(new QName("AddFolder");
Object result = call.invoke(new Object[] {new Long(21), new Long(146),
new String("My new folder")});
```

1.4 Other Helpful Source Files

For additional information about ICEcore Web services, there are a few additional source files that can facilitate your understanding of Web services and product data structures.

This file contains most of the methods that implement the facade:

`/ssf/main/src/com/sitescape/team/remoting/impl/AbstractFacade.java`

This file implements the portions of the facade that use Apache Axis code, especially code managing the sending and receiving of attachments:

`/ssf/main/src/com/sitescape/team/remoting/ws/FacadeImpl.java`

If you want to review the code that builds the workspace tree (along with some helper classes), then you can review this file:

`/ssf/main/src/com/sitescape/team/web/tree/WsDomTreeBuilder.java`

1.5 Notes about Specific Messages

The use of some messages is less intuitive than for other messages. This section provides additional information for those messages and includes the following subsections:

1.5.1 Adding Folders and the Binder Configuration Identifier

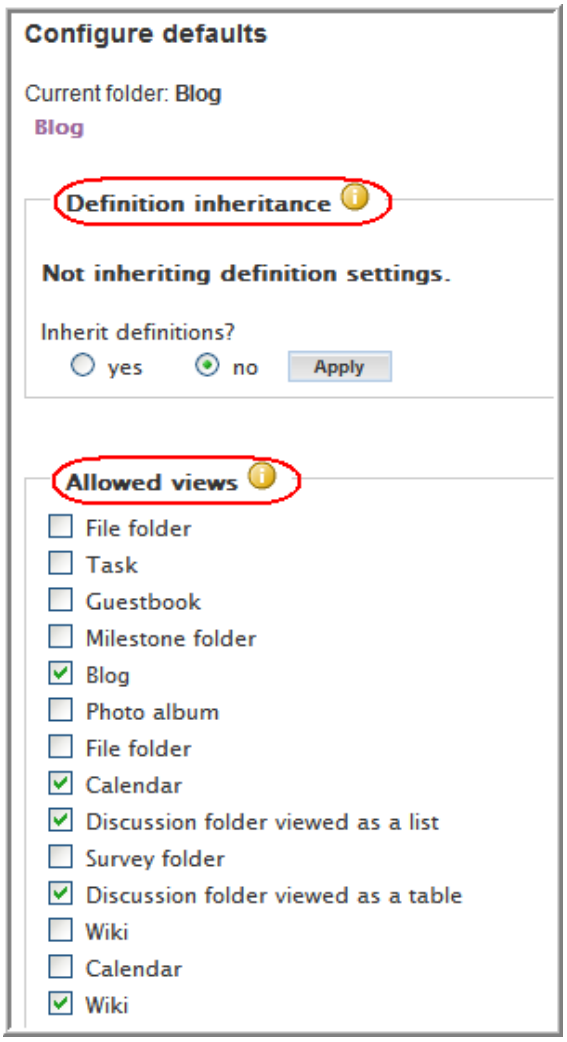
When adding a folder (addFolder (page 27)), you need to specify a binder configuration identifier, which identifies the template used to configure a folder of a particular type. For example, the blog-folder template specifies settings used to configure a new blog folder.

To review the blog-folder template:

1. Click *Manage workspace and folder templates* in the administration portlet.
2. In the "Currently defined templates" section, click *Blog*.

3 Click *Manage this target* > *Configure*.

ICEcore displays a page, the top of which appears as follows:



Here is the complete list of the configuration settings available in the template:

- Definition inheritance
- Allowed views
- Default view
- Default entry types
- Workflow associations
- Allowed workflows

At the time of this writing, ICEcore does not provide a message that you can use to retrieve the binder configuration identifier for a particular type of folder. Instead, to obtain the binder configuration identifier for the folder you want to create, do the following in the UI:

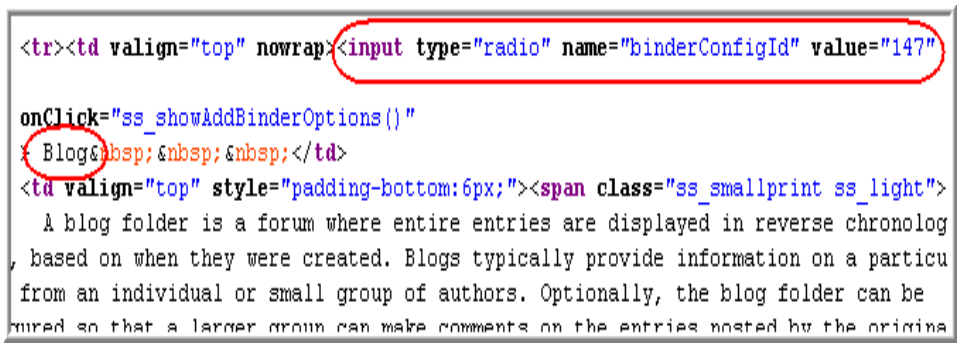
1 View any workspace or folder.

2 Click *Manage* > *Add folder*.

3. While viewing the *Add new folder* page, use your browser to view the HTML source code for the page.
4. Search for the type of folder you want to create (for example, discussion, blog, calendar, and so on).
5. In the `input` HTML tag that creates the radio button for that type of folder, note the `name="binderConfigId"` and `value="nnn"` pair of tag elements.

   The number specified by the `value` element is the binder configuration identifier of the folder you want to create.

Here is an example of the binder configuration information for a blog folder, as found in the HTML source for the *Add new folder* page:



1.5.2 Adding Items and the Definition Identifier

When creating entries (addFolderEntry (page 28)), ICEcore requires a definition identifier, which maps to the definition for the type of entry you want to create. You work with definitions when using the designers in the administration portlet.

The easiest way to work with definition identifiers is to specify `null` for this value. When you specify `null`, ICEcore automatically applies the definition identifier for the default entry type of the folder in which you are creating a new entry. For example, you want to create an entry in a blog folder. If you pass `null` as the definition identifier, ICEcore automatically applies the definition identifier for a blog entry, which is the default entry for a blog folder.

As another option, you can use the `getDefinitionListAsXML` message to get information about all definitions. Then, you can parse the XML string for the definition identifier for the type of entry you want to create.

In the XML string, this is part of a definition for a standard ICEcore item:

```
<definition name="_appItemType"
id="402883b9114739b301114754e8120008"...>
```

To obtain the *`_appItemType`* for the item you want to create and to obtain its corresponding definition identifier:

1. Sign in to ICEcore as a site administrator.
2. In the administration portlet, click the plus sign (+) next to *Form and view designers*.
3. Click the category of the item you want to create using the Web services (for example, click *Entry designer* if you want your Web service call to create an entry).
4. Click the plus sign (+) next to *Entry definitions*.

5. Locate the title of the item you want to create (for example, *Blog entry*), and note the identifier in parentheses next to the title (for example, *_blogEntry*).
6. In the XML returned by `getDefinitionListAsXML`, parse the string until you find the type of item you want to create (for example, `name="_blogEntry"`).
7. Continue parsing that same element until you locate the `id` attribute.

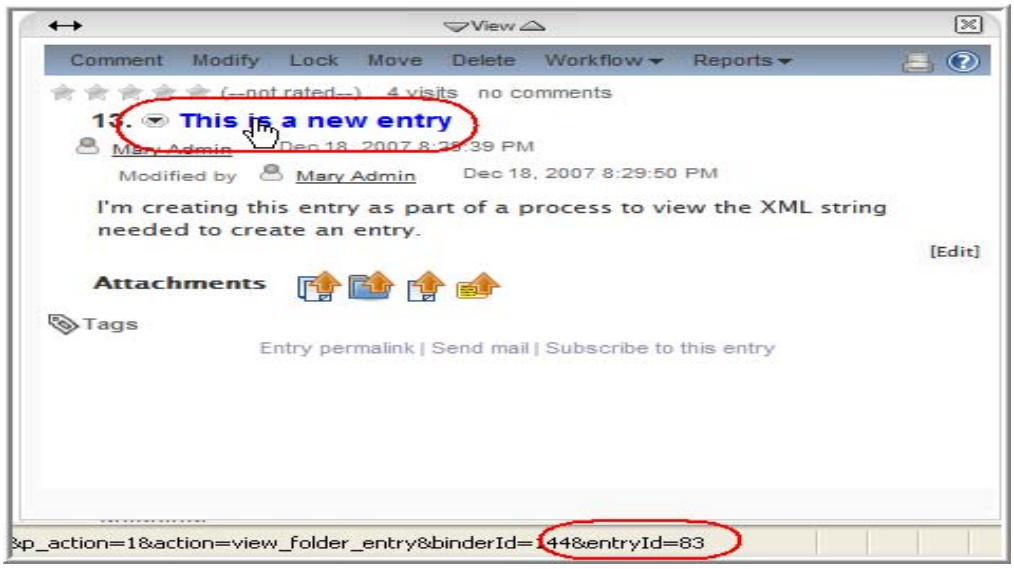
   Retrieve the definition identifier (for example, `id="402883b9114739b301114754e8120008"`).

1.5.3 Tips for All Messages that Add and Modify Entries

When you add (addFolderEntry (page 28)) or modify an entry (modifyFolderEntry (page 57)), you must pass a string of XML containing elements for the entry you want to create or modify. Because ICEcore XML does not adhere to a schema, you need to review a copy of a complete entry to understand the structure of the XML string that you need to specify as a parameter to these messages.

To review the structure of XML required to add or modify an entry, do the following:

1. In the ICEcore UI, create an entry of the type you want your Web services client to be able to create or modify, and specify a value for every element on the creation form.
2. Mouse over the title of the created entry and note the entry identifier in URL displayed in your browser:



   The entry identifier in the previous graphic is 120.
3. Either use the `printFolderEntry` command for `wsclient.bat` or use the `getFolderEntryAsXML` message to view the XML contents of the entry you just created in the UI.
4. Use the XML information you retrieved as a template for the string you need to pass to create a new entry or to modify one.

1.5.4 Attaching Files

In ICEcore, attachments are files that are associated with an entry. An entry can have more than one attached file.

Using Web services, an attachment is a file exchanged in conjunction with a message being passed between the client and server. ICEcore recognizes only the first file attachment to a message being sent to the server and ignores all other attachments.

To attach more than one file to an entry in ICEcore, you must use the `uploadFolderFile` message multiple times (uploadFolderFile (page 66)). So, to attach 17 files to an ICEcore entry, you must use `uploadFolderFile` 17 times. Your client source code establishes where in the file system it finds or places files used as attachments to messages.

The `uploadFolderFile` message requires that you pass a data item name. This identifier maps to the value specified in the `name` attribute of the `input` HTML tag used to upload the file; this value is also used in a `hidden` HTML tag that communicates values between the HTML form and the ICEcore database.

To upload a file into the standard form element used to contain attachments, specify `ss_attachFile` as the data item name. If you are uploading files into a custom form element, follow the instructions in the previous section for viewing the XML of a folder entry (Section 1.5.3, "Tips for All Messages that Add and Modify Entries," on page 16), obtain the name of the custom form element containing attached files, and specify it when adding more attachments.

1.5.5 Fetching Attachments

When you use `getFolderEntryAsXML` to obtain information about an entry (getFolderEntryAsXML (page 38)), you use a boolean parameter to indicate if you want the entry's attachments. If you specify that you do want the attachments, your client establishes where on its system it wishes to place the attached files.

1.5.6 Adding Calendar Entries

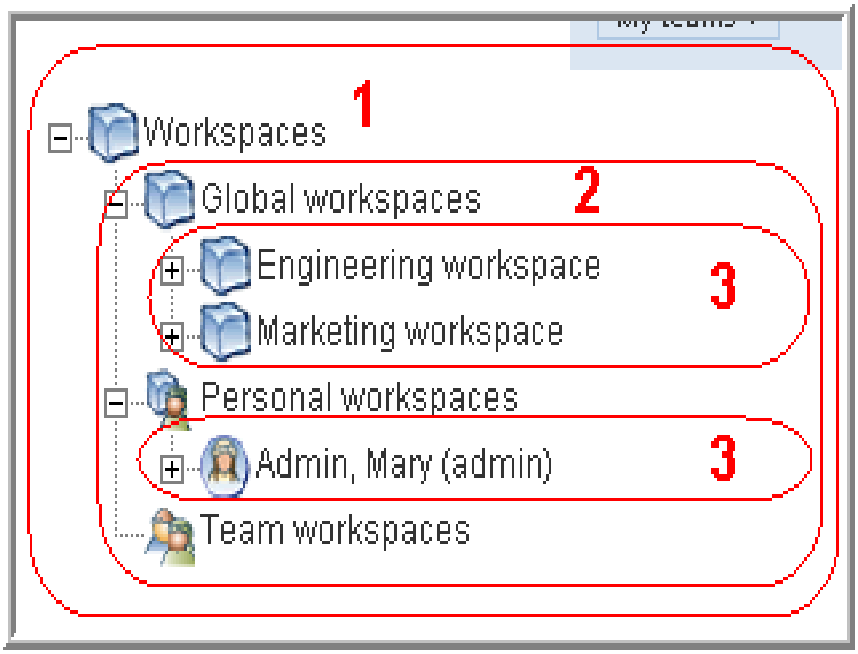
When you pass the `uploadCalendarEntries` message to the server (uploadCalendarEntries (page 65)), the Web services framework uses both an XML formatted string of iCal data passed as the second parameter to the message (`<doc><entry>iCal data</entry></doc>`), and an iCal file that might or might not be passed as an attachment to the message. Although you can split the source data across both the parameter and the attachment, clients generally use one or the other to specify entry data. So, if you want the iCal file attachment to be the source of data for calendar entries, then pass an empty XML document as the second parameter to `uploadCalendarEntries` (`<doc></doc>`).

NOTE: The `uploadCalendar` command in the `wsclient.bat` batch file accepts two required parameters and an optional third parameter. The second parameter is a file containing XML that specifies iCal data. The third, optional parameter is an iCal formatted file. Both files must be located in the same directory as `wsclient.bat`. Again, if you want the iCal file to be the only source of data for newly created entries, place an empty XML document in the file specified as the second command parameter.

1.5.7 Binder Pages and getWorkspaceTreeAsXML

When you use `getWorkspaceTreeAsXML` to obtain information about the hierarchical workspace tree (getWorkspaceTreeAsXML (page 42)), ICEcore returns XML formatted information about nodes in the tree, within the levels of the hierarchy you specify. Each node in the tree is a binder, which is typically a place (a workspace or folder). Sometimes, the XML element returned for a node is called a page.

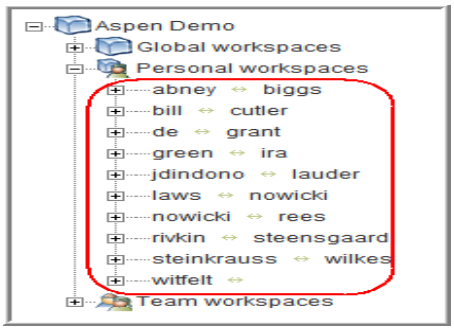
The following page shows the workspace tree, which is expanded to show five levels of the workspace hierarchy.



Each of the depicted workspaces and folders are nodes in the workspace tree. The *Workspaces* workspace is the only binder at level 1. Level 2 binders include *Global workspaces*, *Personal workspaces*, and *Team workspaces*. The only depicted binder at level 3 is the *Corporate web site* binder. Level 4 binders include folders and the *December 2008 redesign* workspace. And the *Calendar* binder is located at level 5. If a binder has a plus sign next to it (for example, both the *Global workspaces* and *Personal workspaces* binders are preceded by plus signs), then it means that there are hierarchy levels of binders that are not yet being displayed in the UI.

If you use `getWorkspaceTreeAsXML` to fetch one level of the tree starting at the *Workspaces* node, ICEcore returns information about *Global workspaces*, *Personal workspaces*, and *Team workspaces*.

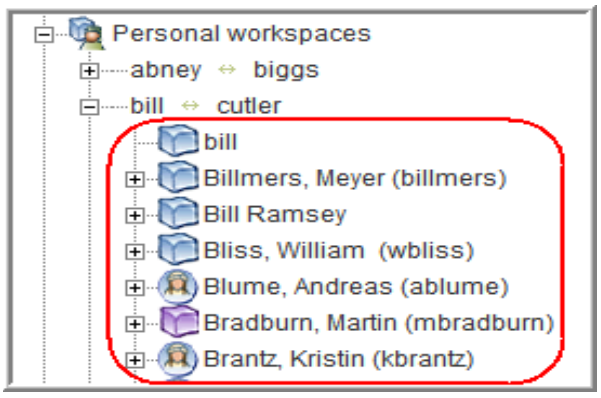
As mentioned, some nodes in the tree are pages, which are shown in here:



The `/ssf/web/docroot/WEB-INF/classes/config/ssf.properties` file contains a property called `wsTree.maxBucketSize`, which, by default, is set to `25`. This means that the maximum number of subworkspaces or subfolders allowed is 25. If a folder or workspace has more subplaces, ICEcore creates virtual buckets called pages. Each line in the previous graphic (*admin <--> raymond* and *starkweather <--> white*) corresponds to a page. The *Personal workspaces* workspace has two pages.

When you use `getWorkspaceTreeAsXML` to retrieve information about nodes in the workspace tree, it can return more than one hierarchical level as you specified, unless it encounters a page. To expand the tree beyond a page, you must call getWorkspaceTreeAsXML again, pass the binder identifier of the page, and pass the number of levels beyond the page you wish to retrieve.

Consider the following:



The *starkweather//white* page contains workspaces. The workspaces listed (*Starkweather, John*, *Tolliver, Michael*, *Waters, Jon*, and so on) are one level beyond the page.

When you receive page information as a node in the workspace tree, you receive `page` and `tuple` attributes. For example, `page="1"` and `tuple="jane_adams//rhonda_hernandez"`. To obtain information about the contents of this page, you need to specify the identifier of the page's parent, the number of hierarchy levels you want expanded, and a concatenation of the page number and tuple values, as shown in this example:

```
getWorkspaceTreeAsXML 24 3 "1//jane_adams//rhonda_hernandez"
```

This code begins at binder number 24, accesses page number 1, and returns 3 hierarchical levels of data for all users between Jane Adams and Rhonda Hernandez.

Given the structure of the ICEcore pages and how Web services returns tree data, it is easiest to retrieve page data in this way. However, if you choose, you can actually retrieve paged tree data

regardless of page number. To do this, specify any page number (ICEcore actually ignores it), and specify a tuple in the correct order in which it appears in the tree, even if the set of users crosses pages. ICEcore returns hierarchical information for all users in between the tuple values. However, if the number of returned nodes exceeds the value specified in the `wsTree.maxBucketSize` property (by default, 25 users), ICEcore pages the data.

Finally, if you want to see all tree information without any page specifications, specify `-1` as the value of the hierarchy levels you want returned.

1.6 Extending ICEcore Web Services

Given that ICEcore is Open Source software, you have the source code that implements our Web services, and you can extend it. However, we invite you to operate within the spirit of an Open Source community by participating in the ICEcore online community, sharing your code with others, and working with the Novell Engineers to incorporate your Web services extensions into the base product. In this way, you make the product and community stronger, and you avoid doing work that might need to be reworked in future versions of ICEcore because of engineering changes.

Of course, whether you participate in the community or upgrade to future versions of the software is up to you. Regardless of your decision, ICEcore includes an example that provides a structure that enables users of all versions of our software to be able to extend our Web services in the most optimal way, minimizing work that you would need to do to maintain the extensions for every upgrade.

ICEcore includes an extended Web services example, which adds the `getBinderTitle` message and the `getBinderTitle` command to the `wsclient.bat` sample client. The source code for the extension is located in this directory and in its subdirectories:

`/ssf/samples/extendedws`

This directory contains the `readme.txt` file, which provides cursory directions for establishing the extension. Those instructions are repeated here and elaborated upon.

To implement the extension:

1. Use a command-line window to go to the `/extendedws` directory.
2. Execute this command:

   ```
   > ant deploy
   ```
3. In the installation directories (not the source code), open this file for editing:

   `/`*`icecore installation dir`*`/webapps/ssf/WEB-INF/server-config.wsdd`

   Replace the string `com.sitescape.team.remoting.ws.JaxRpcFacade` with the string `com.sitescape.team.samples.extendedws.server.JaxRpcFacade2`.

   If you upgrade to a new version of the software, you need to repeat this step to re-implement your Web services extensions.
4. In the same set of directories, open this file for editing:

   `/`*`install dir`*`/webapps/ssf/WEB-INF/context/applicationContext.xml`

   Replace the string `com.sitescape.team.remoting.ws.FacadeImpl` with the string `com.sitescape.team.samples.extendedws.server.FacadeImpl2`.

   Also, replace the string `com.sitescape.team.remoting.Facade` with the string `com.sitescape.team.samples.extendedws.server.Facade2`.

If you upgrade to a new version of the software, you need to repeat this step to re-implement your Web services extensions.

5 Restart the c server.

6 To test the new Web services message, view this page in a browser window:

`http://local host and port/ssf/ws/Facade?wsdl`

The `getBinderTitle` message should now appear.

To test the additional command in the `wsclient.bat` file (Windows only), `cd` to this directory:

`/ssf/samples/remotingclient`

Execute this command:

```
> ant zip
```

And test the newly enabled batch-file command:

```
> wsclient getBinderTitle 1
```

Both the new message and command accept a binder identifier as a parameter and return the textual title of the binder.

1.7 Migrating from Forum to ICEcore

The legacy product of ICEcore is the SiteScape Forum product. To assist with migrating data from SiteScape Forum to an installation of ICEcore, Novell developed a set of Web services.

Although this section provides guidance about migrating, the task is complex and requires the active assistance of the ICEcore support team. This is especially true for workflow migration. For more information, please contact the support team and arrange to receive consultation as you perform this task.

1.7.1 Sequence of Migration Operations

Some operations require the previous execution of other operations. For example, migrating an entry requires that you have already migrated the folder. As another example, a workflow process requires that you have already migrated user and group names, so that these names can be applied to its access control.

Here are notes regarding the sequence of operations:

- Migrate users and groups, and create personal workspaces early in the process.

  Use either LDAP or portal administration to establish the Forum users in ICEcore.

  You need the existence of personal workspaces to be able to migrate sub-workspaces and child folders. Also, migrating workflow and some types of custom commands requires that your users be established in ICEcore first.

  The Forum term “custom command” maps to “custom view and form” in ICEcore.

- Generally, migrate parents before children you wish to create.

  Examples include migrating parent workspaces before its child folders, and migrating entries before migrating attached files.

Using SiteScape Forum, a “forum” maps to a “folder” in ICEcore, a “reply” maps to a “comment” in ICEcore, and the process of “attaching a file” maps to the Web services phrase “adding a folder file.”

- Migrate binders before setting their ownership, team members, and access control.

  The Forum items “workspaces and folders” map to the ICEcore Web services term of “binders.” The Forum term “access control” maps to “membership.” Also, the Web services term “function” is equivalent to the term “roles” in the UI for ICEcore.

- Migrate custom commands before creating entries.

  The custom command migration process cannot be done using only Web services (see Section 1.7.5, “Migrating Custom Commands and Workflow,” on page 23, for more information).

- Migrate workflow processes before migrating entries.

  First, the workflow-migration process cannot be done using only Web services (see Section 1.7.5, “Migrating Custom Commands and Workflow,” on page 23, for more information). Second, any entry that is currently in a workflow state requires the presence of the workflow definition in ICEcore.

- After migrating custom commands and workflow, you can migrate workflow associations for specific folders.

- Finalizing operations include indexing folders and synchronizing any mirrored folders that you created.

  Remember that migrated entries do not appear in the UI until you index the folders containing these entries.

  The ICEcore UI does not begin to mirror the files on the drive until someone manually synchronizes them. The Web services call is equivalent to a manual synchronization in the UI.

1.7.2 Migration Overwrite Operations

Two operations require that you perform the operation for all items using one call to the message; they do not allow you to perform the operation incrementally on subsets of items, using multiple calls to the message. If you call these messages sequentially for subsets of the information, each successive call erases the established data from the previous call.

The messages that require you to perform the operation for all items using only one call are:

- **`setDefinitions`:** Associates entry types with workflow processes (see setDefinitions (page 58)).
- **`setFunctionMembership`:** Sets access control for a workspace or folder (see setFunctionMembership (page 59)).

1.7.3 Migrating Users

Migrating users requires two steps:

1. Use either LDAP or the *Import profiles* administration-portlet tool to add your Forum users to ICEcore.
2. Use the `addUserWorkspace` message to add personal workspaces for the new ICEcore users.

Migrating custom commands and workflow involve additional work in regard to users. See Section 1.7.5, “Migrating Custom Commands and Workflow,” on page 23, for more information.

1.7.4 Migrating Files

If you have a small number of files to migrate to ICEcore, you can use the `migrateFolderFile` message (see migrateFolderFile (page 51)).

However, most Forum installations include a significant number of files, and those files might be large. To improve performance, you should strongly consider using the `migrateFolderFileStaged` message.

Staging involves moving all of the files from SiteScape Forum to the server running the ICEcore installation. Although the files can be located using any folder hierarchy on the server, a convenient way to migrate files is to unzip the Forum hidden directory onto the ICEcore server machine and to work within that existing folder hierarchy from Forum. After placing the files on the ICEcore server, the `migrateFolderFileStaged` message takes files from the staging area and migrates them into the ICEcore installation.

Here are the steps needed to migrate files:

1. Establish a directory on the ICEcore server machine where you want to place the Forum files.
2. Make the three required changes to the `ssf.properties` and `ssf-ext.properties` files. This action indicates the location of the staging directory. (See the installation guide for more information about the these files.)

   Multiple Forum file versions are separate files in the staged area. Call the migrateFolderFileStaged message once for each version of the file, using the same filename for each call but specifying a different path. This method creates versioned files in ICEcore.
3. Copy the Forum files onto the ICEcore server, using the specified staging directory as your top directory.
4. Use the `migrateFolderFileStaged` message to migrate the files into the ICEcore installation.

   This command attaches files to an existing entry. Also, it accepts as one of its arguments a relative path, which traverses the subfolders beneath the designated staging directory.

See migrateFolderFileStaged (page 53), for more information.

1.7.5 Migrating Custom Commands and Workflow

Migrating custom commands and workflow require tasks beyond the scope of using only Web services calls. It is highly recommended that you work closely with the ICEcore support team while completing these tasks.

These are the general steps needed to migrate custom commands and workflow processes:

1. Migrate your Forum users to ICEcore.
2. Use the `getAllPrincipalsAsXML` message to get a list of the user identifiers for the newly created ICEcore users.
3. Create a mapping file that maps ICEcore user identifiers to Forum usernames.

4. Run a Tcl script—which uses the mapping file—to generate an XML file of workflow information.
5. Import the workflow XML file into ICEcore.
6. Create another mapping file, which maps workflow identifiers in ICEcore to Forum workflow names.
7. Run a Tcl script—which uses the second mapping file—to generate an XML file of custom command information.

   Some custom commands are associated with workflow processes. Because of this, the mapping file of workflow information is necessary.
8. Import the custom command XML into ICEcore.

NOTE: This process migrates custom commands created using Forum's user interface. It does not migrate template-based custom commands. To migrate template-based custom commands, use the ICEcore entry designer and any necessary JSPs to recreate the command.

Web Services Message Reference 2

This topic provides alphabetized reference pages for Web services messages provided by the Novell® product called ICEcore.

NOTE: This topic documents the Web services supported for ICEcore Version 1.0.3. As of the next major release of this product, all of these messages will be deprecated in favor of new messages; these older messages will continue to function and will be maintained.

The following are conventions used in this reference section:

What you see	What it means
Click the *Add a team workspace* button. Click the *Getting Started* link.	Items that are clickable on the page, programming variables, or syntax parameters are presented in *italic* font.
Blog summary - Provides a.... **Note:** Remember that....	Defined terms in a list, note headers, section headers on a reference page, and list items on a reference page are presented in bold font.
Type `status`, then press Enter. Open the `ManagerGuide.pdf` file. Use the `open_db` routine with its `lock` parameter.	Text that you must type, file names, commands, command options, routines, Web services messages, and parameters are presented in `Courier` font when occurring in a body of text.
[page]	Optional syntax parameters are enclosed in brackets ([ ]).
..., paramSyntax1 | paramSyntax2,...	Required parameters that accept two or more optional syntaxes are separated by the vertical-line character.
(V1—V1.0.3)	The versions of ICEcore that support the Web services message (“all versions between Version 1.0 through Version 1.0.3”)

NOTE: All examples in this reference section use Apache Axis run-time library methods that specify Web service messages and their argument lists. If you are not using Apache Axis, map the Apache methods to those you are using to implement your Web service calls.

For ICEcore Version 1.0, regarding messages shown in the product source code, the `search` message is under development and subject to change or deletion at any time. Do not use this message in your client applications.

Web service messages contained in this reference section are used by the Windows based clients provided in the ICEcore sources in the `/ssf/samples/remotingclient` folder. With the exception of `uploadCalendar` (uploadCalendarEntries (page 65)), use the same parameters for the batch-file command that you use for the corresponding Web service message.

The following table maps the `wsclient.bat` command name to its corresponding, linked Web services message, which is documented in this reference section:

wsclient.bat command	Web services message
addEntry	addFolderEntry (page 28)
addFolder	addFolder (page 27)
addReply	addReply (page 30)
[none]	addUserWorkspace (page 32)
indexBinder	indexFolder (page 44)
listDefinitions	getDefinitionListAsXML (page 36)
migrateBinder	migrateBinder (page 45)
migrateEntry	migrateFolderEntry (page 49)
migrateReply	migrateReply (page 55)
migrateFile	migrateFolderFile (page 51)
migrateFileStaged	migrateFolderFileStaged (page 53)
migrateWorkflow	migrateEntryWorkflow (page 47)
modifyEntry	modifyFolderEntry (page 57)
printAllPrincipals	getAllPrincipalsAsXML (page 33)
printDefinition	getDefinitionAsXML (page 34)
printDefinitionConfig	getDefinitionConfigAsXML (page 35)
printFolderEntry	getFolderEntryAsXML (page 38)
printFolderEntries	getFolderEntriesAsXML (page 37)
printPrincipal	getPrincipalAsXML (page 39)
printTeamMembers	getTeamMembersAsXML (page 40)
printTeams	getTeamsAsXML (page 41)
printWorkspaceTree	getWorkspaceTreeAsXML (page 42)
setDefinitions	setDefinitions (page 58)
setFunctionMembership	setFunctionMembership (page 59)
setFunctionMembershipInherited	setFunctionMembershipInherited (page 61)
setOwner	setOwner (page 62)
setTeamMembers	setTeamMembers (page 63)
synchronize	synchronizeMirroredFolder (page 64)
uploadCalendar	uploadCalendarEntries (page 65)
uploadFile	uploadFolderFile (page 66)

addFolder

Adds a folder to the workspace-tree hierarchy. (V1—V1.0.3)

Syntax

public long addFolder(long *parentBinderId*, long *binderConfigId*, String *title*);

Description

The `addFolder` message adds a folder to the workspace and folder hierarchy.

Parameters and Return Value

parentBinderId

The identifier of the workspace or folder that is to contain the new folder.

binderConfigId

The identifier that maps to the default configuration for the folder you want to create.

title

A string providing a title for the new entry.

return_value

The binder identifier of the newly created folder.

Example

```
call.setOperationName(new QName("addFolder"));
Object result = call.invoke(new Object[] {new Long(21), new
Long(146), new String("My new folder")});
```

This code creates a new subfolder to the container whose binder identifier is `21`, gives the folder a configuration identifier of 146 (on our test installation, this corresponds to a discussion folder), and establishes the title of the new folder as *My new folder*. The container whose binder identifier is `21` can be either a workspace or folder.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.5.1, "Adding Folders and the Binder Configuration Identifier," on page 13

addFolderEntry

Adds an entry to a folder. (V1—V1.0.3)

Syntax

public long addFolderEntry(long *folderId*, String *definitionId*, String *inputDataAsXML*, String *attachedFileName* | null);

Description

The `addFolderEntry` message adds an entry to a folder.

Parameters and Return Value

folderId

The binder identifier of the folder that is to contain the new entry.

definitionId

The 32-character, hexadecimal identifier that maps to the type of entry to be created (for example, some default entry types are topic, file, blog, wiki, and calendar).

The easiest way to work with definition identifiers for entries is to specify `null` for this value. When you specify `null`, ICEcore automatically applies the definition identifier for the default entry type of the folder in which you are creating a new entry. For example, by default, you want to create an entry in a blog folder. If you pass `null` as the definition identifier, ICEcore automatically applies the definition identifier for a blog entry.

As another option, you can use the `getDefinitionConfigAsXML` message to get information about all definitions. Then, you can parse the XML string for the definition identifier of the type of entry you want.

inputDataAsXML

A string of XML containing the values needed to create an entry of your desired type.

Use the ICEcore UI to create a complete entry of the type you want this Web services message to create, note the entry identifier, and then use the `getFolderEntryAsXML` message to return XML for the entry. Then, use the returned XML as a template for this parameter. (See Section 1.5.3, “Tips for All Messages that Add and Modify Entries,” on page 16, for more information.)

attachedFileName

The name of the file you wish to attach to the new entry. This is an optional parameter. The file must be located in the directory in which the client code executes.

return_value

The entry identifier for the newly created entry.

Examples

```
call.setOperationName(new QName("addFolderEntry"));
Object result = call.invoke(new Object[] {new Long(21), new
String("402883b90cc53079010cc539bf260002"), s, filename},
filename);
```

This code creates a new entry in the folder whose binder identifier is `21`; the specified entry-definition identifier maps to a discussion topic. The variable `s` contains XML elements needed by ICEcore to create the entry. The new entry includes the attached file whose filename is specified by the value of the `filename` variable.

```
call.setOperationName(new QName("addFolderEntry"));
Object result = call.invoke(new Object[] {new Long(21), new
String("402883b90cc53079010cc539bf260002"), s, null});
```

This code produces the same effect as the last example, except that it does not attach a file.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.5.2, "Adding Items and the Definition Identifier," on page 15
- Section 1.5.3, "Tips for All Messages that Add and Modify Entries," on page 16
- getFolderEntryAsXML (page 38)
- getDefinitionConfigAsXML (page 35)

addReply

Adds a new comment to an entry or comment. (V1.0.3)

Syntax

public long addReply(long *folderId*, long *parentEntryId*, String *definitionId*,
String *inputDataAsXML*, String *attachedFileName* | null);

Description

The `addReply` message adds a new comment to an entry or to an existing comment.

Parameters and Return Value

folderId

The binder identifier of the folder containing the entry or comment to which you want to apply the new comment.

parentEntryId

The entry identifier for the entry or comment to which you want to apply the comment.

definitionId

The 32-character, hexadecimal identifier that maps to the type of comment to be created.

You can use the `getDefinitionListAsXML` message to get metadata for all definitions. Then, you can parse the XML string for the definition identifier of the type of comment you want.

inputDataAsXML

A string of XML containing the values needed to create a comment of your desired type.

Use the ICEcore UI to create a complete comment of the type you want this Web services message to create, note the entry identifier, and then use the `getFolderEntryAsXML` message to return XML for the entry. Then, use the returned XML as a template for this parameter. (See Section 1.5.3, “Tips for All Messages that Add and Modify Entries,” on page 16, for more information.)

attachedFileName

The name of the file you wish to attach to the new comment. This is an optional parameter. The file must be located in the directory in which the client code executes.

return_value

The entry identifier of the newly created comment.

Example

```
call.setOperationName(new QName("addReply"));
Object result = call.invoke(new Object[] {new Long(21), new
Long(45), null, s, null});
```

This code creates a new comment in the folder whose binder identifier is `21`, and applies it to the entry or comment whose entry identifier is `45`. The first `null` value instructs ICEcore to use the default comment type for the folder. The variable `s` contains XML elements needed by ICEcore to create the comment. Because of the final `null` value, the new comment does not include an attached file.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.5.2, "Adding Items and the Definition Identifier," on page 15
- Section 1.5.3, "Tips for All Messages that Add and Modify Entries," on page 16
- getFolderEntryAsXML (page 38)
- getDefinitionListAsXML (page 36)

addUserWorkspace

Adds a new personal workspace. (V1.0.3)

Syntax

public long addUserWorkspace (long *userId*);

Description

The `addUserWorkspace` message adds a new personal workspace to the workspace hierarchy.

The primary purpose of this message is to assist with migrating data from SiteScape Forum to ICEcore. By default using ICEcore, the creation of the personal workspace occurs when someone first uses the portal software to sign in with a username and password. If you want to migrate Forum information as sub-content to a personal workspace in ICEcore, use this message before creating the sub-content.

Parameters and Return Value

userId

The identifier for the user for whom you want to create the personal workspace

return_value

The binder identifier of the newly created personal workspace.

Example

```
call.setOperationName(new QName("addUserWorkspace"));
Object result = call.invoke(new Object[] {new Long(21)});
```

This code creates a new personal workspace.

See Also

- Section 1.7.1, "Sequence of Migration Operations," on page 21
- Section 1.7.3, "Migrating Users," on page 22

getAllPrincipalsAsXML

Returns summary information for users and groups. (V1—V1.0.3)

Syntax

public String getAllPrincipalsAsXML(int *firstRecord*, int *maxRecords*);

Description

The `getAllPrincipalsAsXML` message returns XML elements that provide summary information about registered users and defined groups. You can use this message to identify a particular user by name or other data, obtain an identifier for a particular user, and then use the `getPrincipalAsXML` message to gather a finer level of information about that person.

Parameters and Return Value

firstRecord

The index of the first record whose user or group information you want to obtain. The index for the first principal in the system is `1`.

maxRecords

The maximum number of user and group records whose information should be returned.

You can use the previous parameter and this parameter in subsequent calls to `getAllPrincipalsAsXML` to process data for sets of users and groups at a time (for example, 50 at a time, or 100 at a time).

return_value

A string containing the XML elements providing information about the requested set of users and groups.

Example

```
call.setOperationName(new QName("getAllPrincipalsAsXML"));
Object result = call.invoke(new Object[] {new Integer(100), new
Integer(50)});
```

This code requests information for users and groups starting with the record number 100 and including up to 50 records.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- getPrincipalAsXML (page 39)

getDefinitionAsXML

Returns information about one definition. (V1—V1.0.3)

Syntax

public String getDefinitionAsXML(String *definitionId*);

Description

The `getDefinitionAsXML` message returns an XML string containing information about one definition. You work with definitions using the designers in the administration UI.

For example, if you pass one of the definition identifiers for an entry type listed in the `addFolderEntry` reference page, ICEcore returns information about the definition for that entry.

As an alternative, you can use the `getDefinitionConfigAsXML` message to obtain all definitions in ICEcore and then parse the larger string for the definition information you want.

Parameter and Return Value

definitionId

The identifier of the definition whose information you want. Definitions are maintained using the designers in the administration UI, and define the components of an object in ICEcore.

return_value

A string of XML whose elements provide information about the components of an object in ICEcore.

Example

```
call.setOperationName(new QName("getDefinitionAsXML"));
Object result = call.invoke(new Object[] {new
String("402883b9114739b301114754e8120008")});
```

This code requests XML-formatted information about the definition for a wiki entry.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, “Web Services Message Reference,” on page 25)
- Section 1.5.2, “Adding Items and the Definition Identifier,” on page 15
- addFolderEntry (page 28)
- getDefinitionConfigAsXML (page 35)

getDefinitionConfigAsXML

Returns information about all configuration definitions. (V1—V1.0.3)

Syntax

public String getDefinitionConfigAsXML( );

Description

The `getDefinitionConfigAsXML` message returns information about all configuration definitions. The configuration information does not include workflow or template definitions. You can uses the returned information to extract the definition identifier for a given entry type to use in a subsequent call to `addFolderEntry`.

Return Value

return_value

A string of XML whose elements describe all configuration definitions.

Example

```
call.setOperationName(newQName("getDefinitionConfigAsXML"));
Object result = call.invoke();
```

This code obtains information about all configuration settings.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- addFolderEntry (page 28)

getDefinitionListAsXML

Returns metadata for all definitions in the installation. (V1.0.3)

Syntax

public String getDefinitionListAsXML ();

Description

The `getDefinitionListAsXML` message returns metadata for all definitions in the installation. This metadata includes information such as the definition name and identifier.

When using other Web services messages that require a definition identifier, you can use this message, parse the XML for the name (discussion, blog, calendar, comment), and obtain the 32-character, hexadecimal identifier that maps to the desired object.

Return Value

return_value

A string of XML whose elements contain metadata for all definitions in the installation.

Example

```
call.setOperationName(new QName("getDefinitionListAsXML"));
Object result = call.invoke();
```

This code obtains metadata for all definitions in the installation.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)

getFolderEntriesAsXML

Returns a string containing XML providing summary information about entries in a folder. (V1—V1.0.3)

Syntax

public String getFolderEntriesAsXML(long *folderId*);

Description

The `getFolderEntriesAsXML` message returns XML elements containing summary information about each entry in the specified folder.

Parameter and Return Value

folderId

The binder identifier of the folder containing the entries for which you want information.

return_value

A string containing XML elements containing summary information for each entry in the folder specified by `folderId`.

Example

```
call.setOperationName(new QName("getFolderEntriesAsXML"));
Object result = call.invoke(new Object[] {new Long(21)});
```

This code returns a string containing XML information for all of the entries in the folder whose binder identifier is `21`.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)

getFolderEntryAsXML

Returns information about one entry in a folder. (V1—V1.0.3)

Syntax

public String getFolderEntryAsXML(long *folderId*, long *entryId*, boolean *includeAttachments*);

Description

The `getFolderEntryAsXML` message returns XML whose elements provide information about one entry in a folder.

Parameters and Return Value

folderId

The binder identifier of the folder containing the entry whose information you want.

entryId

The identifier of the entry whose information you want.

includeAttachments

A boolean value that indicates whether you want ICEcore to return the entry's attachments. The client program is responsible for placement of attachment files on its local system.

return_value

A string containing XML elements for the requested entry.

Example

```
call.setOperationName(new QName("getFolderEntryAsXML"));
Object result = call.invoke(new Object[] {new Long(21), new
Long(34), new Boolean.FALSE});
```

This code returns XML that includes information contained in entry number 34 in the folder whose identifier is 21. Because of the value of the last parameter, ICEcore does not place the entry's file attachments in the client program's source directory.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.5.5, "Fetching Attachments," on page 17

getPrincipalAsXML

Returns information about one user or group. (V1—V1.0.3)

Syntax

public String getPrincipalAsXML(long *binderId*, long *principalId*);

Description

The `getPrincipalAsXML` message returns XML whose elements provide information about one registered user or defined group.

Parameters and Return Value

binderId

The binder identifier of the principal's parent workspace. The information returned by `getAllPrincipalsAsXML` includes the binder number of this containing workspace.

principalId

The identifier that maps to the user or group for which you want to gather information.

return_value

A string containing XML elements whose elements provide information about the specified user or group.

Example

```
call.setOperationName(new QName("getPrincipalAsXML"));
Object result = call.invoke(new Object[] {new Long(2), new
Long(25)});
```

This code returns information about a user or group, whose parent workspace has a binder identifier of `2` and whose principal identifier is `25`.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- getAllPrincipalsAsXML (page 33)

getTeamMembersAsXML

Returns information about all team members assigned within a workspace or folder. (V1—V1.0.3)

Syntax

public String getTeamMembersAsXML(long *binderId*);

Description

The `getTeamMembersAsXML` message returns XML that names members of a team assigned within the specified workspace or folder.

Parameter and Return Value

binderId

The binder identifier of the workspace or folder for which you want information about team members. The `getTeamsAsXML` message returns information about all workspaces and folders that have assigned teams.

return_value

A string containing XML elements describing team members for the specfiied place.

Example

```
call.setOperationName(new QName("getTeamMembersAsXML"));
Object result = call.invoke(new Object[] {new Long(23));
```

This code returns an XML string whose elements describe all of the team members assigned in the workspace or folder associated with the binder identifier of `23`.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- getTeamsAsXML (page 41)

getTeamsAsXML

Returns information about all workspaces and folders that have assigned teams. (V1—V1.0.3)

Syntax

public String getTeamsAsXML( );

Description

The `getTeamsAsXML` message returns an XML string providing information about all workspaces and folders that have assigned teams. You can use this message to obtain the list of places that have assigned teams, note a binder number of a particular place, and then use the `getTeamMembersAsXML` message to obtain the list of team members for that place.

Return Value

return_value

An XML string whose elements describe workspaces and folders that have assigned teams.

Example

```
call.setOperationName(new QName("getTeamsAsXML"));
Object result = call.invoke();
```

This code returns information about all places in the `ICEcore` installation that have assigned teams.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- getTeamMembersAsXML (page 40)

getWorkspaceTreeAsXML

Returns information needed to construct the ICEcore workspace and folder tree. (V1—V1.0.3)

Syntax

public String getWorkspaceTreeAsXML(long *binderId*, int *levels*, String *page*);

Description

The `getWorkspaceTreeAsXML` message returns XML elements needed to construct the requested portion of the ICEcore workspace tree.

Parameters and Return Value

binderId

The binder identifier of the starting node of the returned portion of the hierarchy. The top workspace in the ICEcore tree has a binder identifier of `1`.

levels

The number of hierarchical levels down from the node specified by `binderId` that you want to include in the returned information. The value `-1` indicates that you want all subsequent levels.

page

A parameter used to expand pages of binders. When you specify a valid page identifier, ICEcore expands the page by the levels indicated in the `levels` parameter.

If you do not want to expand pages using this call, pass `null` as this parameter.

The Web-services overview topic contains more detailed information about working with pages (Section 1.5.7, “Binder Pages and getWorkspaceTreeAsXML,” on page 18).

return_value

A string containing XML elements needed to construct each node within the requested levels of the workspace hierarchy.

Example

```
call.setOperationName(new QName("getWorkspaceTreeAsXML"));
Object result = call.invoke(new Object[] {new Long(1), new
Integer(3), null});
```

This code returns a string containing XML information for the first three levels of the workspace hierarchy. The following depicts these levels using default workspace titles:

```
Level 1: Workspaces
     Level 2: Global, Personal, and Team workspaces
          Level 3: Children of Global, Personal, and Team
```

The children of *Global workspaces*, *Personal workspaces*, and *Team workspaces* can be either workspaces or folders.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, “Web Services Message Reference,” on page 25)
- Section 1.5.7, “Binder Pages and getWorkspaceTreeAsXML,” on page 18

indexFolder

Indexes a folder. (V1.0.3)

Syntax

public void indexFolder(long *folderId*);

Description

The `indexFolder` message indexes a folder.

The primary use of this message is to index data after you migrate it from SiteScape Forum into ICEcore. (The migration messages transfer the data but do not index it.)

Parameter

folderId

The binder identifier of the folder you want to index.

Example

```
call.setOperationName(new QName("indexFolder"));
Object result = call.invoke(new Object[] {new Long(21)});
```

This indexes the folder whose binder identifier is `21`.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21

migrateBinder

Creates a new workspace or folder while preserving SiteScape Forum data. (V1.0.3)

Syntax

public long migrateBinder (long *parentId*, String *definitionId*, String *inputDataAsXML*, String *creator*, Calendar *creationDate*, String *modifier*, Calendar *modificationDate*);

Description

The `migrateBinder` message creates a workspace or folder in ICEcore that preserves values from a SiteScape Forum installation (for example, the name of the person who created the item in Forum, the Forum creation date, the person who last modified the item in Forum, and the date of the last modification in Forum).

Parameters and Return Value

parentId

The binder identifier of the parent of the newly created workspace or folder.

definitionId

The 32-character, hexadecimal identifier that maps to the type of workspace or folder to be created.

You can use the `getDefinitionListAsXML` message to get metadata for all definitions. Then, you can parse the XML string for the definition identifier of the type of workspace or folder you want to create.

inputDataAsXML

A string of XML supplying the elements and values needed to construct the workspace or folder you want to create.

creator

A string containing the username of the person who created the corresponding workspace or folder in the Forum installation.

creationDate

Calendar data specifying the date when the corresponding workspace or folder was created in Forum.

modifier

A string containing the username of the person who last modified the corresponding workspace or folder in Forum.

modificationDate

Calendar data specifying the date when the corresponding workspace or folder was modified in Forum.

return_value

The binder identifier of the newly created workspace or folder.

Example

```
call.setOperationName(new QName("migrateBinder"));
Object result = call.invoke(new Object[] {new Long(21), def, input,
new String("JSmith"), createcal, new String("JGarces"), modcal});
```

This code creates a new binder determined by the definition in the `def` variable (use the `getDefinitionListAsXML` message to obtain the correct string for your binder type), and the binder will be a child of the binder whose identifier is `21`. The `input` variable contains an XML string, properly formatted for your binder type, which ICEcore uses to create binder content. The remaining four parameters provide names (literals) and dates (the `createcal` and `modcal` variables) for the creation and last modification of the corresponding item in the Forum installation.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21
- getDefinitionListAsXML (page 36)

migrateEntryWorkflow

Associates an entry with a workflow process while preserving SiteScape Forum data. (V1.0.3)

Syntax

public void migrateEntryWorkflow (long *binderId*, long *entryId*, String *definitionId*, String *startState*, String *modifier*, Calendar *modificationDate*);

Description

The `migrateEntryWorkflow` message associates a workflow process with an entry in ICEcore, while preserving values from a SiteScape Forum installation (for example, the state to which the entry should be set, the person who last changed workflow state in Forum, and the date of the last state change in Forum).

Parameters and Return Value

binderId

The binder identifier of the folder that contains the entry to which you want to associate a workflow process.

entryId

The entry identifier of the entry to which you want to associate a workflow process.

definitionId

The 32-character, hexadecimal identifier that maps to the workflow-process definition.

Before using this message, you must replicate the Forum workflow processes in ICEcore.

startState

The current state of the ICEcore entry (which would reflect its last set set in Forum).

modifier

A string containing the username of the person who last changed the workflow process in Forum.

modificationDate

Calendar data specifying the date when the workflow process was last changed in Forum.

Example

```
call.setOperationName(new QName("migrateEntryWorkflow"));
Object result = call.invoke(new Object[] {new Long(21), new
Long(45), String("ptoProcess"), String("PTO Request"), new
String("JGarces"), modcal});
```

This code associates the `ptoProcess` workflow process with the entry whose identifier is `45` and which is located in a folder whose binder identifier is `21`. The entry should be placed in the

`PTO Request` state. The message also provides the name of the person who last changed the workflow state in Forum and the date when that state change occurred.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21
- Section 1.7.5, "Migrating Custom Commands and Workflow," on page 23

migrateFolderEntry

Creates a new folder entry while preserving SiteScape Forum data. (V1.0.3)

Syntax

public void migrateFolderEntry (long *binderId*, String *definitionId*, String *inputDataAsXML*, String *creator*, Calendar *creationDate*, String *modifier*, Calendar *modificationDate*);

Description

The `migrateFolderEntry` message creates a folder entry in ICEcore that preserves values from a SiteScape Forum installation (for example, the name of the person who created the item in Forum, the Forum creation date, the person who last modified the item in Forum, and the date of the last modification in Forum).

When creating entries within a file folder in ICEcore, use this message to create the entry, and then use either `migrateFolderFile` or `migrateFolderFileStaged` to attach the file to the entry.

Parameters and Return Value

binderId

The binder identifier of the folder to contain the new entry.

definitionId

The 32-character, hexadecimal identifier that maps to the type of entry to be created.

The easiest way to work with definition identifiers for entries is to specify `null` for this value. When you specify `null`, ICEcore automatically applies the definition identifier for the default entry type of the folder in which you are creating a new entry. For example, by default, you want to create an entry in a blog folder. If you pass `null` as the definition identifier, ICEcore automatically applies the definition identifier for a blog entry.

As another option, you can use the `getDefinitionListAsXML` message to get metadata for all definitions. Then, you can parse the XML string for the definition identifier of the type of workspace or folder you want to create.

inputDataAsXML

A string of XML supplying the elements and values needed to construct the type of entry you want to create.

creator

A string containing the username of the person who created the corresponding entry in the Forum installation.

creationDate

Calendar data specifying the date when the corresponding entry was created in Forum.

modifier

A string containing the username of the person who last modified the corresponding entry in Forum.

modificationDate

Calendar data specifying the date when the corresponding entry was modified in Forum.

return_value

The entry identifier of the newly created entry.

Example

```
call.setOperationName(new QName("migrateFolderEntry"));
Object result = call.invoke(new Object[] {new Long(21), def, input,
new String("JSmith"), createcal, new String("JGarces"), modcal});
```

This code creates a new entry of the type determined by the definition in the `def` variable (use the `getDefinitionListAsXML` message to obtain the correct string for your entry type), and the new entry is to be located in the binder whose identifier is `21`. The `input` variable contains an XML string, properly formatted for your entry type, which ICEcore uses to create entry content. The remaining four parameters provide names (literals) and dates (the `createcal` and `modcal` variables) for the creation and last modification of the corresponding entry in the Forum installation.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.5.3, "Tips for All Messages that Add and Modify Entries," on page 16
- Section 1.7.1, "Sequence of Migration Operations," on page 21
- getDefinitionListAsXML (page 36)
- migrateFolderFile (page 51)
- migrateFolderFileStaged (page 53)

migrateFolderFile

Attaches a file to an entry while preserving SiteScape Forum data. (V1.0.3)

Syntax

public void migrateFolderFile (long *binderId*, long *entryId*, String *fileUploadDataItemName*, String filename, String *modifier*, Calendar *modificationDate*);

Description

The `migrateFolderFile` message attaches a file to a folder entry in ICEcore that preserves values from a SiteScape Forum installation (for example, the person who last modified the item in Forum, and the date of the last modification in Forum).

Parameters and Return Value

binderId

The binder identifier of the folder that contains the entry to which you want to attach a file.

entryId

The entry identifier of the entry to which you want to attach the file.

fileUploadDataItemName

The internal-use name used by the database to identify the file as an element of an entry.

For example, a Forum custom command allowed for uploading different files into a single entry that served different functions, such as an expense report, a meeting presentation, and so on. These custom file uploads have associated internal-use names that are different than the reserved internal-use name applied to standard file entries or standard attachments.

If you are migrating to a folder file, specify `upload` as an argument to this parameter to make this attachment the primary file for the entry.

filename

The name of the file to be attached to the entry.

modifier

A string containing the username of the person who last modified the corresponding file in Forum.

modificationDate

Calendar data specifying the date when the corresponding file was modified in Forum.

Example

```
call.setOperationName(new QName("migrateFolderFile"));
Object result = call.invoke(new Object[] {new Long(21), new
```

```
Long(45), String("_budgetReport"), String("budget-report.xls"),
new String("JGarces"), modcal});
```

This code attaches the `budget-report.xls` file to the entry whose identifier is `45` and is located in a folder whose binder identifier is `21`. The internal-use name that maps to the file as an element in the entry is `_budgetReport`. The message also provides the name of the person who modified the file in Forum and the date when that modification occurred.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21
- Section 1.7.4, "Migrating Files," on page 23

migrateFolderFileStaged

Locates a locally stored file, and attaches it to an entry while preserving Forum data. (V1.0.3)

Syntax

public void migrateFolderFileStaged (long *binderId*, long *entryId*,
String *fileUploadDataItemName*, String filename, String *stagedFileRelativePath*, String *modifier*,
Calendar *modificationDate*);

Description

The `migrateFolderFileStaged` accesses a Forum file that has been copied locally on the ICEcore server as a way to streamline the transfer of files, avoiding transferring them over the Internet. The message then attaches the file to a folder entry in ICEcore that preserves values from a SiteScape Forum installation (for example, the person who last modified the item in Forum, and the date of the last modification in Forum).

Parameters and Return Value

binderId

The binder identifier of the folder that contains the entry to which you want to attach a file.

entryId

The entry identifier of the entry to which you want to attach the file.

fileUploadDataItemName

The internal-use name used by the database to identify the file as an element of an entry.

For example, a Forum custom command allowed for uploading different files into a single entry that served different functions, such as an expense report, a meeting presentation, and so on. These custom file uploads have associated internal-use names that are different than the reserved internal-use name applied to standard file entries or standard attachments.

If you are migrating to a folder file, specify `upload` as an argument to this parameter to make this attachment the primary file for the entry.

filename

The name of the file to be attached to the entry.

stagedFileRelativePath

The relative path specification, beginning with the staging area designated in the `ssf.properties` and `ssf-ext.properties` files on the ICEcore server. (See the installation guide for more information about these files.)

Although the files can be present in any folder structure within the staging area, one streamlined way to approach this task is to unzip the Forum hidden directory into the staging area. Then, use this parameter to specify the relative path through the hidden folder structure to the location of the file to be attached to the entry in ICEcore.

modifier

A string containing the full name of the person who last modified the corresponding file in Forum.

modificationDate

Calendar data specifying the date when the corresponding file was modified in Forum.

Example

```
call.setOperationName(new QName("migrateFolderFileStaged"));
Object result = call.invoke(new Object[] {new Long(21), new
Long(45), String("_budgetReport"), String("budget-report.xls"),
String("hidden/ssf/myworkspace/myforum/4567849"), new
String("JGarces"), modcal});
```

To locate the file, ICEcore begins with the defined staging folder and then applies the relative path `hidden/ssf/myworkspace/myforum/456789`.This code attaches the `budget-report.xls` file to the entry whose identifier is `45` and is located in a folder whose binder identifier is `21`. The internal-use name that maps to the file as an element in the entry is `_budgetReport`. The message also provides the name of the person who modified the file in Forum and the date when that modification occurred.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21
- Section 1.7.4, "Migrating Files," on page 23

migrateReply

Creates a new comment while preserving SiteScape Forum data. (V1.0.3)

Syntax

public void migrateReply (long *binderId*, long *parentId*, String *definitionId*,
String *inputDataAsXML*, String *creator*, Calendar *creationDate*, String *modifier*,
Calendar *modificationDate*);

Description

The `migrateReply` message creates a comment in ICEcore that preserves values from a SiteScape Forum installation (for example, the name of the person who created the item in Forum, the Forum creation date, the person who last modified the item in Forum, and the date of the last modification in Forum).

Parameters and Return Value

binderId

The binder identifier of the folder that will contain the new comment.

parentId

The binder identifier of the entry or comment to which you want to apply the new comment.

definitionId

The 32-character, hexadecimal identifier that maps to the type of comment to be created.

You can use the `getDefinitionListAsXML` message to get metadata for all definitions. Then, you can parse the XML string for the definition identifier of the type of comment you want to create.

inputDataAsXML

A string of XML supplying the elements and values needed to construct the type of comment you want to create.

creator

A string containing the username of the person who created the corresponding reply in the Forum installation.

creationDate

Calendar data specifying the date when the corresponding reply was created in Forum.

modifier

A string containing the username of the person who last modified the corresponding reply in Forum.

modificationDate

Calendar data specifying the date when the corresponding reply was modified in Forum.

return_value

The entry identifier of the newly created comment.

Example

```
call.setOperationName(new QName("migrateReply"));
Object result = call.invoke(new Object[] {new Long(21), new
Long(45), def, input, new String("JSmith"), createcal, new
String("JGarces"), modcal});
```

This code creates a new comment of the type determined by the definition in the `def` variable (use the `getDefinitionListAsXML` message to obtain the correct string for your comment type). The new comment is to be located in the binder whose identifier is 21, and applied to an entry or comment whose identifier is 45. The `input` variable contains an XML string, properly formatted for your comment type, that ICEcore uses to create comment content. The remaining four parameters provide names (literals) and dates (the `createcal` and `modcal` variables) for the creation and last modification of the corresponding reply in the Forum installation.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.5.3, "Tips for All Messages that Add and Modify Entries," on page 16
- Section 1.7.1, "Sequence of Migration Operations," on page 21

modifyFolderEntry

Modifies a single entry. (V1—V1.0.3)

Syntax

public void modifyFolderEntry(long *folderId*, long *entryId*, String *inputDataAsXML*);

Description

The `modifyFolderEntry` message modifies one entry in a folder.

Parameters and Return Value

folderId

The binder identifier of the folder that contains the entry to be modified.

entryId

The identifier of the entry to be modified.

inputDataAsXML

A string of XML containing the values needed to modify the entry.

return_value

None.

Example

```
call.setOperationName(new QName("modifyFolderEntry"));
Object result = call.invoke(new Object[] {new Long(21), new
Long(43), s});
```

This code modifies entry 43 in the folder whose binder ID is `21`. The variable `s` contains XML elements needed by ICEcore to modify the contents of the entry.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.5.3, "Tips for All Messages that Add and Modify Entries," on page 16

setDefinitions

Establishes workflow-entry associations for a folder. (V1.0.3)

Syntax

public void migrateEntryWorkflow (long *binderId*, String[] *definitionIds*,
String[] *workflowAssociations*);

Description

The `setDefinitions` message uses two arrays to associate workflow identifiers with entry identifiers for a folder. (ICEcore associates identifiers in the first element of both arrays, the second element of both arrays, the third, and so on.)

When an entry is associated with a workflow process, creation of an entry of that type automatically places the entry into the initial state of the workflow process.

NOTE: This message is an overwrite operation, setting all workflow associations for the folder; you cannot use repeated calls to this message to set associations incrementally. So, set all of the workflow associations for the folder with one call.

Parameters and Return Value

binderId

The binder identifier of the folder in which you want to associate entry and workflow identifiers.

definitionIds

An array of entry identifiers.

workflowAssociations

An array of workflow identifiers.

Before using this message, you must replicate the Forum workflow processes in ICEcore.

Example

```
call.setOperationName(new QName("setDefinitions"));
Object result = call.invoke(new Object[] {new Long(21), entries,
workflows});
```

This code passes two array variables, `entries` and `workflows`. ICEcore uses the corresponding elements in both arrays to create entry-workflow associations for the folder whose binder identifier is `21`.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)

setFunctionMembership

Applies access-control settings to a folder or workspace. (V1—V1.0.3)

Syntax

public void setFunctionMembership(long *binderId*, String *inputDataAsXML*);

Description

The `setFunctionMembership` message provides access-control settings for folder or a workspace. The term function is analogous to a role in the user interface (UI).

The primary use of this message is to establish access-control settings when migrating workspaces and folders from Forum to ICEcore. You must ensure that you have migrated Forum user and group names to ICEcore that are required for your access-control settings.

NOTE: This message is an overwrite operation, setting all function memberships for the folder or workspace; you cannot use repeated calls to this message to set memberships incrementally. So, set all memberships for the workspace or folder with one call.

Parameters and Return Value

binderId

The binder identifier of the folder or workspace for which you want to set access control.

inputDataAsXML

A string of XML containing the values needed to set access control. Here is an example of XML that sets the visitor function:

```
<workAreaFunctionMemberships>
<workAreaFunctionMembership>
<property name="functionName">__role.visitor</property>
<property name="memberName">jGarces</property>
<property name="memberName">sChen</property>
<proprty name="members">1,2,3</property>
</workAreaFunctionMembership>
   .
   .
   .
</workAreaFunctionMemberships>
```

To obtain the `functionName` value:

1. Sign in as a site administrator for ICEcore.
2. In the administration portlet, click *Configure role definitions*.
3. Click any item (for example, *Participant*).
4. Note or copy the identifier in the *Role Name* text box (for example, `__role.participant`). This identifier begins with a double underscore (_).

You can pass either user or group names (for example, `jGarces` or `sChen`) or user or group identifiers (for example, `1`, `2`, `3`). ICEcore reserves the identifiers `-1` for the workspace or folder owner, and `-2` for a team member.

Example

```
call.setOperationName(new QName("setFunctionMembership"));
Object result = call.invoke(new Object[] {new Long(21), s});
```

This code uses the content of the XML string s to establish access-control settings for the folder or workspace whose binder identifier is 21.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21

setFunctionMembershipInherited

Establishes inheritance as the access-control mechanism for a folder or workspace. (V1.0.3)

Syntax

public void setFunctionMembershipInherited(long *binderId*, boolean *inherit*);

Description

The `setFunctionMembershipInherited` message allows you to establish that a folder or workspace is to inherit its access-control settings from the parent binder. The primary purpose of this message is to set inheritance for folders and workspaces that you migrate from Forum.

Parameters and Return Value

binderId

The binder identifier of the folder or workspace for which you want to establish inheritance for its access-control settings.

inherit

A boolean value that determines whether the folder or workspace uses inheritance to establish its access settings.

Example

```
call.setOperationName(new
QName("setFunctionMembershipInherited"));
Object result = call.invoke(new Object[] {new Long(21), new
Boolean.TRUE});
```

This code establishes inheritance as the access-control mechanism for the folder or workspace whose binder identifier is 21.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21

setOwner

Establishes the owner of a folder or workspace. (V1.0.3)

Syntax

public void setOwner(long *binderId*, long *userId*);

Description

The `setOwner` message allows you to establish an owner for a folder or workspace. The primary purpose of this message is to mirror Forum ownership as you migrate folders and workspaces.

Parameters and Return Value

binderId

The binder identifier of the folder or workspace for which you want to establish ownership.

userId

The user identifier of the person whom you want to be the owner of a folder or workspace.

Example

```
call.setOperationName(new QName("setOwner"));
Object result = call.invoke(new Object[] {new Long(21), new
Long(345)});
```

This code establishes the user whose identifier is `345` as the owner of the folder or workspace whose binder identifier is 21.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21

setTeamMembers

Establishes the membership of a team for a folder or workspace. (V1.0.3)

Syntax

public void setTeamMembers(long *binderId*, String[] *memberNames*);

Description

The `setTeamMembers` message establishes the members of a team for a folder or workspace.

Parameters and Return Value

binderId

The binder identifier of the folder or workspace for which you want to establish team membership.

memberNames

An array containing the names of all team members for the folder or workspace.

Example

```
call.setOperationName(new QName("setTeamMembers"));
Object result = call.invoke(new Object[] {new Long(21), users});
```

This code establishes each username in the array `users` as team members for the folder or workspace whose binder identifier is 21.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21

synchronizeMirroredFolder

Synchronizes the mirrored folder with the folder on the external drive. (V1.0.3)

Syntax

public void synchronizeMirroredFolder(long *binderId*);

Description

The `synchronizeMirroredFolder` message synchronizes a mirrored folder with the corresponding file on the external drive. A new mirrored folder does not synchronize with its external drive until a synchronization occurs manually in the user interface (UI) or using this message.

Parameters and Return Value

binderId

The binder identifier of the mirrored file that you want to synchronize with its external drive.

Example

```
call.setOperationName(newQName("synchronizedMirroredFolder"));
Object result = call.invoke(new Object[] {new Long(21)});
```

This code synchronizes with its external drive the mirrored folder whose binder identifier is 21.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.7.1, "Sequence of Migration Operations," on page 21

uploadCalendarEntries

Creates new calendar entries from a file. (V1—V1.0.3)

Syntax

public void uploadCalendarEntries(long *folderId*, String XMLCalendarData);

Description

The `uploadCalendarEntries` message uses iCal information in an XML string or in an attachment to add entries to a calendar folder.

NOTE: The `uploadCalendar` command in the `wsclient.bat` batch file accepts two required parameters and an optional third parameter. The second parameter is a file containing XML that specifies iCal data. The third, optional parameter is an iCal formatted file. Both files must be located in the same directory as `wsclient.bat`. Again, if you want the iCal file to be the only source of data for newly created entries, place an empty XML document in the file specified as the second command parameter.

Parameters and Return Value

folderId

The binder identifier of the calendar folder that is to contain the new entries.

XMLCalendarData

A string containing XML formatted calendar data (`<doc><entry>iCal data</entry>...</doc>`). If you wish to specify all of your calendar data in an iCal file attached to the message, pass an empty document for this string (`<doc></doc>`).

return_value

None.

Example

```
call.setOperationName(new QName("uploadCalendarEntries"));
Object result = call.invoke(new Object[] {new Long(21), s});
```

This code creates new entries in the calendar folder whose binder ID is `21`. ICEcore uses XML-formatted iCal information contained in the `s` variable to create the new calendar entries.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, "Web Services Message Reference," on page 25)
- Section 1.5.6, "Adding Calendar Entries," on page 17

uploadFolderFile

Attaches a file to an entry to a folder. (V1—V1.0.3)

Syntax

public void uploadFolderFile(long *folderId*, String *entryId*, String *fileUploadDataItemName*, String *attachedfileName*);

Description

The `uploadFolderFile` message attaches a file to an entry in a folder. You can attach only one file at a time; call this message multiple times to attach more than one file to the entry. Files to be attached must be located in the same directory as the executing client.

Parameters and Return Value

folderId

The binder identifier of the folder that contains the entry to which you want to attach a file.

entryId

The identifier of the entry to which you want to attach a file.

fileUploadDataItemName

A string containing the internal identifier for the part of the entry that contains attached files. This identifier maps the `name` attribute of an `input` HTML tag on a form to the ICEcore database; a `hidden` HTML tag communicates this mapping to the server.

The `name` value for the standard entry element containing attached files is `ss_attachFile`. If you want to upload a file into a custom form element you defined using the designers, you need to look up the `name` identifier for that form element (see also `getDefinitionConfigAsXML` or `getFolderEntryAsXML`).

attachedFileName

The name of the file you wish to attach to the new entry. This client is responsible for locating on its local system the file to be used as an attachment.

return_value

None.

Example

```
call.setOperationName(new QName("uploadFolderFile"));
Object result = call.invoke(new Object[] {new Long(21), new
Long(43), new String("ss_attachFile"), filename}, filename);
```

This code attaches a file to entry 43 in the folder whose binder ID is `21`. The name of the file to be attached to the entry is contained in the variable `filename`.

See Also

- The message table for the Windows based `wsclient.bat` program (Chapter 2, “Web Services Message Reference,” on page 25)
- Section 1.5.4, “Attaching Files,” on page 17
- getDefinitionConfigAsXML (page 35)
- getFolderEntryAsXML (page 38)

Documentation Updates

A

This topic describes new and changed sections of this guide, which describes using Web services for the Novell® application ICEcore.

A.1 July 14, 2008

These sections include updates:

Location	Update
Section 1.1.1, “Sample Clients,” on page 8	Added information about downloading source files for ICEcore, which is necessary to run the sample clients.
Section 1.7, “Migrating from Forum to ICEcore,” on page 21	Added overview information about migrating from SiteScape Forum to ICEcore.
addReply (page 30)	Added a new reference page for this message.
addUserWorkspace (page 32)	Added a new reference page for this message.
getDefinitionListAsXML (page 36)	Added a new reference page for this message.
indexFolder (page 44)	Added a new reference page for this message.
migrateBinder (page 45)	Added a new reference page for this message.
migrateEntryWorkflow (page 47)	Added a new reference page for this message.
migrateFolderEntry (page 49)	Added a new reference page for this message.
migrateFolderFile (page 51)	Added a new reference page for this message.
migrateFolderFileStaged (page 53)	Added a new reference page for this message.
migrateReply (page 55)	Added a new reference page for this message.
setFunctionMembership (page 59)	Added a new reference page for this message.
setFunctionMembershipInherited (page 61)	Added a new reference page for this message.
setOwner (page 62)	Added a new reference page for this message.
setTeamMembers (page 63)	Added a new reference page for this message.
synchronizeMirroredFolder (page 64)	Added a new reference page for this message.