

HP Enterprise Collaboration

For the Windows® operating systems

Software Version: 1.1

Integration Guide

Document Release Date: August 2012

Software Release Date: August 2012



Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 2012 Hewlett-Packard Development Company, L.P.

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

<http://h20230.www2.hp.com/selfsolve/manuals>

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support Online web site at:

<http://www.hp.com/go/hpsoftwaresupport>

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

Integration Guide	1
Contents	5
Introduction	8
What is an Adapter?	9
Integrating the Adapter	9
Possible Adapter Deployments	9
Adapter Configuration Content	10
Adapter Configuration Version Control	11
REST Adapter API	12
basicAdapterUrl/rest/config	13
basicAdapterUrl/rest/config/ {configVersion}	13
basicAdapterUrl/rest/config/ {configVersion}/btoObjects?id={btoObjectId} &type={btoObjectType}	13
basicAdapterUrl/rest/config/ {configVersion}/btoObjects?type={btoObjectType} &id={btoObjectId1} ...&id={btoObjectIdn}	13
basicAdapterUrl/rest/config/ {configVersion}/facets/{facetName} ?btoObjectId={btoObjectId} &btoObjectType={btoObjectType}	14
basicAdapterUrl/rest/config/ {version}/btoObjectSearchResult?type={btoObjectType} &max_result={max_result}&search_params	14
basicAdapterUrl/rest/config/ {version}/searchDynamicField/{searchDynamicField}?search_params	15
Adapter Wrapper	16
Java Adapter Wrapper	16
Developing an Adapter using the JAVA Adapter Wrapper	16
Adapter Service Interface	17
String getDefaultConfigXml()	18
Object parseGlobalAdapterConfig (String globalAdapterConfigString)	18
Object parseBtoObjectTypeAdapterConfig (String btoObjectType, String btoObjectAdapterConfigString)	18
Object parseFacetAdapterConfig (String btoObjectType, String facetName, String facetAdapterConfigString)	19
boolean isObjectSupported(BtoObjectDescriptor btoObject, String securityToken, AdapterConfigWrapper adapterConfigWrapper)	19

TableDataModel getFacetData(String faceName, BtoObjectDescriptor btoObject, AdapterConfigWrapper adapterConfigWrapper, String securityToken)	20
String getFacetHtml(TableDataModel tableDataModel, BtoObjectDescriptor btoObject, String facetName, AdapterConfigWrapper adapterConfigWrapper, String baseUrl, String securityToken);	20
String getThumbnailHtml(TableDataModel tableDataModel, BtoObjectDescriptor btoObject, String facetName, AdapterConfigWrapper adapterConfigWrapper, String baseUrl, String securityToken)	21
List<BtoObject> getBtoObjects(String type, List<String> ids, String securityToken, AdapterConfigWrapper adapterConfigWrapper);	22
BtoObjectSearchResult searchBtoObjects(String type, Map<String, Object> parameterMap, int maxResult, String securityToken, AdapterConfigWrapper adapterConfigWrapper);	22
List<String> getSearchFieldValues(String fieldName, Map parameterMap, String securityToken, AdapterConfigWrapper adapterConfigWrapper);	23
Adapter Wrapper Deployment Steps	23
.NET Adapter Wrapper	24
Implementing the .NET Adapter	24
Deployment Steps	25
How to Add an Adapter	25
REST API Schemas	26
Adapter Configuration Schema	27
XSD Structure	27
Sample Code	30
Facet Result Schema	32
Sample Code	33
Context Object Schema	33
Sample Code	34
Search Result Schema	35
Sample Code	35
Adapter JMX Methods	36
URL Launch	37
Authentication	38
Display Inbox URL	38
Search Conversations by Context Objects URL	39

Context Objects XML Structure	40
URL Attachments	41
Sample URL Attachments XML	41
Search Results	41
Compact Mode	44
Compact Mode Functionality	44
JavaScript Integration	46
Implementing JavaScript in Enterprise Collaboration	46
Search According to One Object Id Call	46
Search According to Several Object Id Calls	47
Check Enterprise CollaborationStatus	47
.NET Example for JavaScript Integration	48
Glossary	51

Chapter 1

Introduction

This guide describes how to develop an adapter and implement it in Enterprise Collaboration. The intended audience for this guide are Adapter Developers.

This chapter contains the following sections:

What is an Adapter?	9
Integrating the Adapter	9
Adapter Configuration Content	10
Adapter Configuration Version Control	11

What is an Adapter?

An adapter is a design pattern that translates one interface for a class into a compatible interface. It is also responsible for retrieving data from an external application and transforming it into the required format for Enterprise Collaboration.

In Enterprise Collaboration, an adapter is used to integrate with external applications and to add content. Adapters are also used to retrieve facets and search for context objects. The process of integrating Enterprise Collaboration into external applications is described in this guide.

Enterprise Collaboration can work with many external data source applications. It uses the adapter to retrieve, process and present the external data.

The external application developer is responsible for developing the adapter.

Enterprise Collaboration uses a "code name" approach for all adapters and does not constrain developers to use a common language for all adapters (e.g. query language). This means that each adapter receives the name of the facet to be returned, which prevents the ability to customize the facet outside the adapter. To enable facet customization, put the application query details in the adapter's configuration file.

Integrating the Adapter

This section introduces the adapter development process. After the REST API is implemented, Enterprise Collaboration customers can use it to retrieve and present information from their own data sources.

The customer implements a predefined REST API. This API allows Enterprise Collaboration to retrieve the default configuration, to determine whether the current Context object is supported by the adapter, to search for Context objects and to retrieve the facet data or presentation (or both) for a specific Context object.

Enterprise Collaboration provides default facet presentations for quick integration for cases where the adapter can only support facet data.

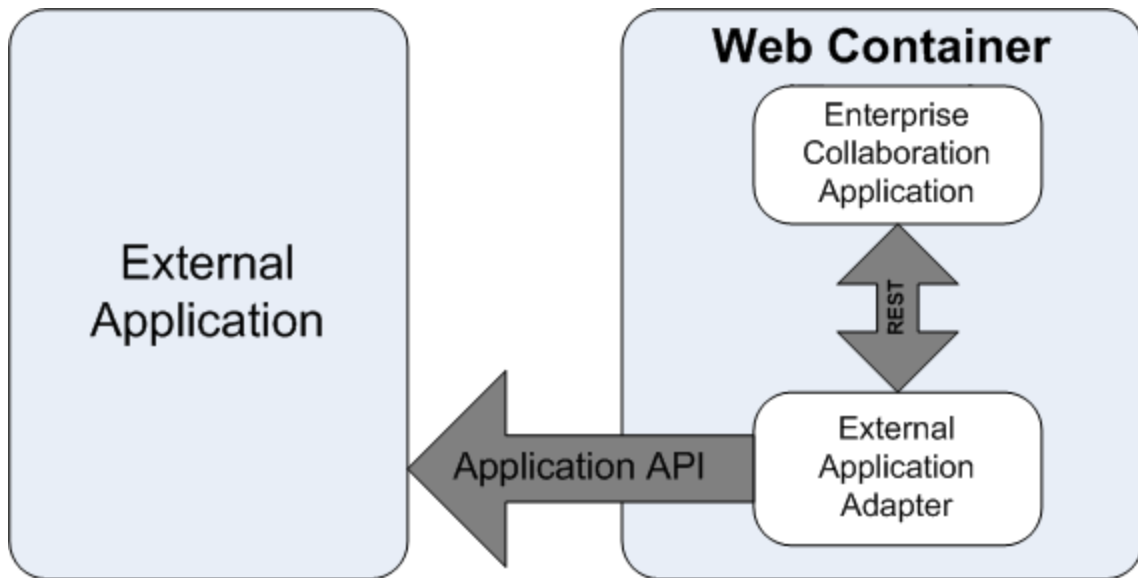
The main purpose of the adapter's configuration file is to enable quick customization of a facet in the field. It is assumed that the administrator has enough knowledge to customize a specific facet in terms of a specific application query language.

Possible Adapter Deployments

The following diagrams show samples of two types of adapter deployments: within the Enterprise Collaboration Web Container, and in an External Application.

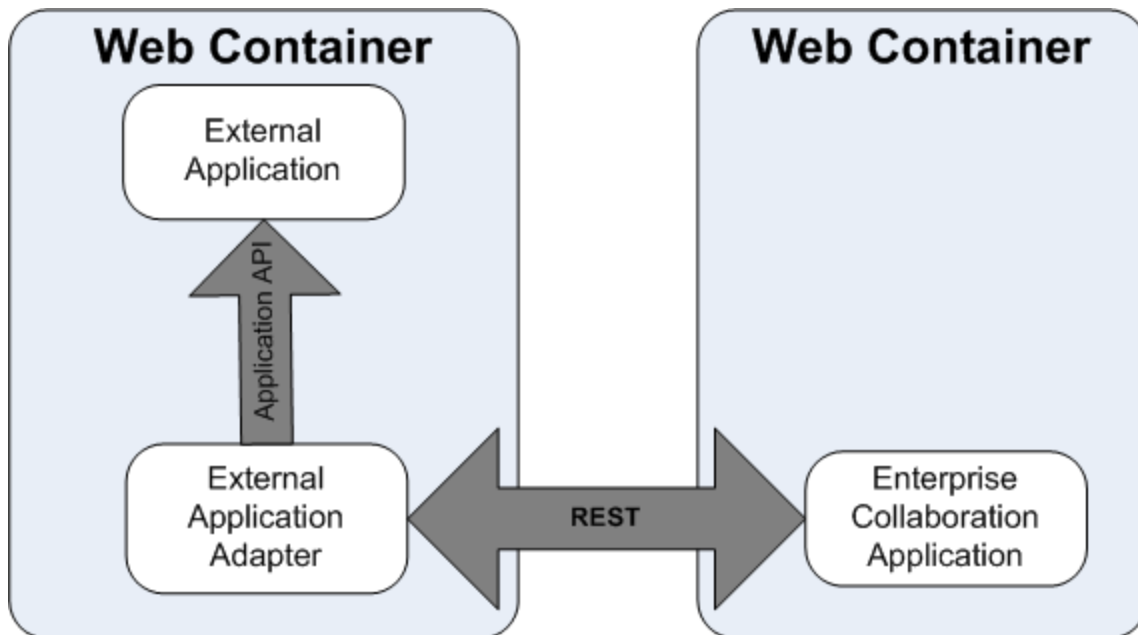
Enterprise Collaboration Web Container Deployment

In this deployment , the developer embeds the adapter into the Enterprise Collaboration installation.



External Application Deployment

In this deployment, the developer installs the adapter at an external location and the Enterprise Collaboration installation calls it from that location.



Adapter Configuration Content

The adapter configuration contains the following information:

- Supported Context object types – the Context object types that the adapter can recognize.
- Supported facets for each Context object type.

- The presentation definition for each facet – can be default or custom. If the case of custom, the adapter must provide the facet html presentation in addition to data. The data and thumbnail html are optional.
- Search information for Context object types that support search. Search information contains the search fields (parameters). The adapter should enable searching the Context objects from the certain type by these field's values. The fields can be numeric, alphanumeric, enumerated or dynamic. Enumerated fields have the static possible values that are defined in adapter configuration. Dynamic fields are like enumerated fields but the possible values are retrieved from the adapter dynamically.
- Adapter proprietary global configuration, i.e. application connection configuration.
- Adapter proprietary configuration per facet, i.e. the mapping for application fields.
- Adapter proprietary configuration per Context object type, i.e. what Application API to call.

For examples of adapter configuration data, see "[REST API Schemas](#)" (on page 26).

Adapter Configuration Version Control

Enterprise Collaboration manages the adapter's configuration versions as follows:

- The adapter provides Enterprise Collaboration with its default configuration using REST API. The first time the adapter is added, Enterprise Collaboration reads the default configuration, processes it, and assigns version number 0 to it.
- To change an adapter configuration, the administrator uses the JMX method.
- When the adapter configuration is updated, Enterprise Collaboration saves it and increments its version number by one.
- Every time Enterprise Collaboration calls the adapter, the adapter receives the current configuration version number as a parameter. If the configuration version differs from the adapter's version, the adapter generates an exception error with the error code `HttpError.Conflict`. In this case, Enterprise Collaboration updates the adapter with the new configuration using REST API and calls the required API again.

Chapter 2

REST Adapter API

This section presents a list of the APIs used to handle the adapter configuration.

basicAdapterUrl/rest/config	13
basicAdapterUrl/rest/config/ {configVersion}	13
basicAdapterUrl/rest/config/ {configVersion}/btoObjects?id={btoObjectId} &type={btoObjectType}	13
basicAdapterUrl/rest/config/ {configVersion}/btoObjects?type={btoObjectType} &id={btoObjectId1} ...&id={btoObjectIdn}	13
basicAdapterUrl/rest/config/ {configVersion}/facets/{facetName} ?btoObjectId={btoObjectId} &btoObjectType={btoObjectType}	14
basicAdapterUrl/rest/config/ {version}/btoObjectSearchResult?type={btoObjectType} &max_result={max_result}&search_params	14
basicAdapterUrl/rest/config/ {version}/searchDynamicField/{searchDynamicField}?search_params	15

basicAdapterUrl/rest/config

Request Method	Description
GET	Returns the default adapter configuration. This method is called after the adapter is added to Enterprise Collaboration. It is also called when the Refresh Configuration method is called from JMX. The returned default configuration is saved in Enterprise Collaboration

basicAdapterUrl/rest/config/ {configVersion}

Request Method	Description
POST	Updates the adapter configuration with the configuration version number provided. The request body contains the new adapter configuration.

basicAdapterUrl/rest/config/ {configVersion} /btoObjects?id={btoObjectId} &type={btoObjectType}

Request Method	Description
HEAD	Returns a response without message body (without error) if the object is supported by the adapter (the adapter can provide facets for this object) or produces an Exception with 404 (Not Found) error code. The Exception with error code 409 (conflict) is produced if the configuration version number is different from the one existed in the adapter.

basicAdapterUrl/rest/config/ {configVersion} /btoObjects?type={btoObjectType} &id={btoObjectId1} ...&id={btoObjectIdn}

Request Method	Description
GET	Returns the required Context objects result according to required configuration version. The Exception with error code 409 (conflict) is thrown if the configuration version number is different from the one existing in the adapter. Each Context object contains the information according to the btoObject.xsd schema.

basicAdapterUrl/rest/config/ {configVersion} /facets/{facetName} ?btoObjectId={btoObjectId} &btoObjectType={btoObjectType}

Request Method	Description
GET	<p>Returns the required facet result according to the required configuration version. The request parameters btoObjectType and btoObjectId describe the Context object. The facet result facet_result.xsd (see "REST API Schemas" (on page 26)) contains:</p> <ul style="list-style-type: none">• TableDataModel A general presentation of facet data in a simple 2D table. This data will be saved in Enterprise Collaboration and can be used for facet comparison or future analysis. If the source application does not support API for retrieving the data, but only supports html presentation, this data is optional and only the custom presentation can be used.• FacetHtmlPresentation A custom html presentation (defined by the adapter) of the facet. It will be ignored if default presenter is used.• ThumbnailHtmlPresentation A custom thumbnail html presentation (defined by the adapter). The thumbnail facet presentation is a small facet presentation that follows the post content. The adapter implementer may use the ThumbnailHtmlPresentation field to indicate important data for the small facet presentation. This field is optional. If the custom presenter is used and this field is empty, "FacetHtmlPresentation" is used to create the thumbnail. If the default presentation is used, it will be ignored in any case. <p>If the configuration version number differs from the one that exists in the adapter, the Exception with error code 409 (conflict) exception is produced</p>

basicAdapterUrl/rest/config/ {version} /btoObjectSearchResult ?type={btoObjectType}&max_result= {max_result}&search_params

Request Method	Description
GET	Returns the Context object search result according to requested search parameters. The max_result parameter defines the maximum

Request Method	Description
	<p>number of objects to return in the search result. The search_params are additional search parameters that are defined by the adapter in the configuration (see "REST API Schemas" (on page 26)).</p> <p>The search result contains the total number of Context objects that match the requested search parameters and Context objects descriptors of the first max_result objects (see "Search Result Schema " (on page 35)).</p> <p>The exception with error code 409 (conflict) exception is produced if the configuration version number differs from the one that exists in the adapter.</p>

basicAdapterUrl/rest/config/ {version}/searchDynamicField/ {searchDynamicField}?search_params

Request Method	Description
GET	<p>Returns the SearchFieldValues that contain possible values of searchDynamicField. search_params contains the values of the previous dynamic fields.</p> <p>The exception with error code 409 (conflict) exception is produced if the configuration version number differs from the one that exists in the adapter.</p>

Each REST request receives the LWSSO token as a query. The adapter may need the LWSSO token for accessing the source application. If the source application also uses the LWSSO framework, it can be used as the URL parameter, or can be decrypted to retrieve the user name.

Chapter 3

Adapter Wrapper

Enterprise Collaboration provides adapter wrappers written in Java/.NET for quick adapter development.

Java Adapter Wrapper

The Java Adapter Wrapper handles two issues:

- **REST API implementation** – The adapter wrapper receives the REST calls and transforms them into the JAVA API of the AdapterService interface. See ["REST Adapter API" \(on page 12\)](#) for details.
- **Adapter configuration parsing and version control** - The adapter wrapper parses the adapter configuration to the AdapterConfigWrapper class.

The **AdapterConfigWrapper** contains facet configuration and parsed adapter proprietary configurations (global, Context object type, facet and search configurations).

The facet configuration contains only the required information for creating the facet data and frees the adapter from redundancy information about presentation configuration.

The adapter wrapper also checks if the current configuration version fits the Enterprise Collaboration adapter configuration version. If not, it sends the required exception.

Developing an Adapter using the JAVA Adapter Wrapper

To create your own adapter using the Adapter Wrapper you should implement the AdapterService interface and define the implementation class as a Spring component.

The adapter wrapper contains the jar files for development and the war file for the adapter deployment.

To start the AdapterService interface implementation, put the adapter-wrapper.jar and lib directory inside the **adapter-wrapper.war** into the **classpath**.

You must put the compiled adapter classes in **adapter-wrapper.war** in **WEB-INF\classes**

To deploy the Adapter Wrapper, follow the steps described in "Setup the Adapter" in the Enterprise Collaboration Installation and Configuration Guide.

Adapter Service Interface

The Adapter Service interface comprises the following methods:

String getDefaultConfigXml()	18
Object parseGlobalAdapterConfig (String globalAdapterConfigString)	18
Object parseBtoObjectTypeAdapterConfig (String btoObjectType, String btoObjectAdapterConfigString)	18
Object parseFacetAdapterConfig (String btoObjectType, String facetName, String facetAdapterConfigString)	19
boolean isObjectSupported(BtoObjectDescriptor btoObject, String securityToken, AdapterConfigWrapper adapterConfigWrapper)	19
TableDataModel getFacetData(String faceName, BtoObjectDescriptor btoObject, AdapterConfigWrapper adapterConfigWrapper, String securityToken)	20
String getFacetHtml(TableDataModel tableDataModel, BtoObjectDescriptor btoObject, String facetName, AdapterConfigWrapper adapterConfigWrapper, String baseUrl, String securityToken);	20
String getThumbnailHtml(TableDataModel tableDataModel, BtoObjectDescriptor btoObject, String facetName, AdapterConfigWrapper adapterConfigWrapper, String baseUrl, String securityToken)	21
List<BtoObject> getBtoObjects(String type, List<String> ids, String securityToken, AdapterConfigWrapper adapterConfigWrapper);	22
BtoObjectSearchResult searchBtoObjects(String type, Map<String, Object> parameterMap, int maxResult, String securityToken, AdapterConfigWrapper adapterConfigWrapper);	22
List<String> getSearchFieldValues(String fieldName, Map parameterMap, String securityToken, AdapterConfigWrapper adapterConfigWrapper);	23

String getDefaultConfigXml()

Input

None.

Output

The default adapter configuration.

Object parseGlobalAdapterConfig (String globalAdapterConfigString)

Input

Parameter	Description
globalAdapterConfigString	The proprietary global adapter configuration that is part of the adapter configuration xml.

Output

Parameter	Description
<Proprietary_Adapter_Data_Structure>	Contains parsed information from the object type adapter configuration. The adapter will receive this object as part of the AdapterConfigWrapper.

Object parseBtoObjectTypeAdapterConfig (String btoObjectType, String btoObjectAdapterConfigString)

Input

Parameter	Description
btoObjectType	The Context object type for which the current configuration is defined.
btoObjectAdapterConfigString	The proprietary Context object type adapter configuration for certain Context object type that is part of the adapter configuration xml.

Output

Parameter	Description
<Proprietary_Adapter_Data_Structure>	Contains parsed information from the global adapter configuration. The adapter will receive this object as part of AdapterConfigWrapper.

Object parseFacetAdapterConfig (String btoObjectType, String facetName, String facetAdapterConfigString)

Input

Parameter	Description
btoObjectType	The Context object type for which the current facet configuration is defined.
facetName	The facet name of the current configuration.
facetAdapterConfigString	The proprietary facet adapter configuration for certain Context object type and facet name that is part of the adapter configuration xml.

Output

Parameter	Description
<Proprietary_Adapter_Data_Structure>	Contains parsed information from the facet adapter configuration. The adapter will receive this object as part of the AdapterConfigWrapper parameter.

boolean isObjectSupported(BtoObjectDescriptor btoObject, String securityToken, AdapterConfigWrapper adapterConfigWrapper)

Input

Parameter	Description
btoObject	A Context object for which Enterprise Collaboration wishes to know possible facets. For this purpose it should know if the adapter supports this Context object instance. The Context object contains its ID and Type.
securityToken	The adapter may need the LWSSO token for accessing the source application. It can be used as URL parameter, if the source application also uses the LWSSO framework or can be decrypted to retrieve the user name.
adapterConfigWrapper	The current adapter configuration.

Output

Parameter	Description
<boolean>	<p>True if the adapter supports the current instance of the Context object.</p> <p>False otherwise.</p> <p>Note: If the object is not supported, the adapter wrapper will produce an appropriate exception.</p>

TableDataModel getFacetData(String faceName, BtoObjectDescriptor btoObject, AdapterConfigWrapper adapterConfigWrapper, String securityToken)

Input

Parameter	Description
facetName	Required facet name.
btoObject	<p>A Context object for which the facet data is required.</p> <p>This object contains its ID and Type.</p>
adapterConfigWrapper	The current adapter configuration.
securityToken	The adapter may need the LWSSO token for accessing the source application. It can be used as URL parameter, if the source application also uses the LWSSO framework or can be decrypted to retrieve the user name.

Output

A TableDataModel object for containing basic 2D table and header fields.

String getFacetHtml(TableDataModel tableDataModel, BtoObjectDescriptor btoObject, String facetName, AdapterConfigWrapper adapterConfigWrapper, String baseUrl, String securityToken);

Input

Parameter	Description
tableDataModel	The facet data retrieved by the getFacetData method.
btoObject	A Context object for which the facet html is required.

Parameter	Description
	This object contains its ID and Type.
facetName	Required facet name.
adapterConfigWrapper	The current adapter configuration.
baseUrl	The HTML base URL (all resources should be relative to this value).
securityToken	The adapter may need the LWSSO token for accessing the source application. It can be used as URL parameter, if the source application also uses the LWSSO framework or can be decrypted to retrieve the user name.

Output

The custom html facet presentation. The html format will be used for facet presentation if the custom presentation is defined for the facet in the adapter configuration, otherwise it will be ignored.

Note: In general, the custom html presentation should be rendered from the **tableDataModel**. An exception is when the **getFacetData** method is not supported by the adapter and returns null. In this case the adapter should create the html using the call source application.

String getThumbnailHtml(TableDataModel tableDataModel, BtoObjectDescriptor btoObject, String facetName, AdapterConfigWrapper adapterConfigWrapper, String baseUrl, String securityToken)

Input

Parameter	Description
tableDataModel	The facet data retrieved by the method getFacetData.
btoObject	A Context object for which the facet thumbnail html is required. This object contains its ID and Type.
facetName	The required facet name.
adapterConfigWrapper	The current adapter configuration.
baseUrl	The HTML base URL (all resources should be relative to this value).
securityToken	The adapter may need the LWSSO token for accessing the source application. It can be used as URL parameter, if the source application also uses the LWSSO framework or can be decrypted to retrieve the user name.

Output

The custom html facet presentation. The thumbnail html format will be used for thumbnail facet presentation if the custom presentation is defined for the facet in the adapter configuration, otherwise it will be ignored.

Note: In general, the custom thumbnail html presentation should be rendered from the **tableDataModel**. An exception is when the **getFacetData** method is not supported by the adapter and returns null. In this case, the adapter should create the html using the call source application.

List<BtoObject> getBtoObjects(String type, List<String> ids, String securityToken, AdapterConfigWrapper adapterConfigWrapper);

Input

Parameter	Description
type	Type of the required Context objects.
ids	Ids of required Context objects.
securityToken	The adapter may need the LWSSO token for accessing the source application. It can be used as URL parameter, if the source application also uses the LWSSO framework or can be decrypted to retrieve the user name.
adapterConfigWrapper	The current adapter configuration.

Output

The list of required Context objects, according to given type and ids.

BtoObjectSearchResult searchBtoObjects(String type, Map<String,Object> parameterMap, int maxResult, String securityToken, AdapterConfigWrapper adapterConfigWrapper);

Input

Parameter	Description
type	Type of the required Context objects.
parameterMap	Contains the value of the search fields that are defined in the adapter configuration. The key is the search field name and the value is the array of values that was entered by the user. In the most cases, it is the single value.
maxResult	Defines the maximum number of objects to return in the search result.

Parameter	Description
securityToken	The adapter may need the LWSSO token for accessing the source application. It can be used as URL parameter, if the source application also uses the LWSSO framework or can be decrypted to retrieve the user name.
adapterConfigWrapper	Current adapter configuration.

Output

BtoObjectSearchResult that contains the total number of Context objects that fits the search parameters and the list of first maxResult Context objects (BtoObjectDescriptor) that fits the search parameters.

List<String> getSearchFieldValues(String fieldName, Map parameterMap, String securityToken, AdapterConfigWrapper adapterConfigWrapper);

Input

Parameter	Description
fieldName	The name of the required search field.
parameterMap	Contains the value of the search fields that are defined in the adapter configuration. The key is the search field name and the value is the array of values that was entered by the user. In the most cases, it is the single value.
securityToken	The adapter may need the LWSSO token for accessing the source application. It can be used as the URL parameter, if the source application also uses the LWSSO framework or can be decrypted to retrieve the user name.
adapterConfigWrapper	The current adapter configuration.

Output

List of possible values for required search field.

Adapter Wrapper Deployment Steps

In order to deploy the Adapter Wrapper, follow the steps below:

1. Change the LWSSO initString value to the same one that is defined in Enterprise Collaboration installation:
 - a. In the adapter **war** file, go to **WEB-INF\classes**.
 - b. Open the **adapterlwssofmconf.xml** and replace the initString value (the value for initializing symmetric encryption) with the correct value. The initString is defined in **<EC_**

Installation_Folder>\servers\server-0\webapps\diamond\WEB-INF\classes\diamond\wssofmconf.xml.

2. Copy the resulting **war** file in the directory **<EC_Installation_Folder>\servers\server-0\webapps**. The name of the **war** file should be the same as the adapter name.

Note: In order to prevent network speed issues, copy the adapter **war** file to the temporary directory in the target server. Then after the deployment, move it from the temporary directory to the directory **<EC_Installation_Folder>/servers/server-0/webapps**.

3. Add the basic adapter URL using JMX as follows:
 - a. Go to **<EC_Application_URL>/diamond/jmx-console**
 - b. Select: **Diamond > Diamond adapter config JMX service**.
 - c. Invoke the **addAdapterUrl** method with the following parameters:
 - o Adapter name
 - o Adapter basic URL. The adapter URL should be in the format **{local}/adapter_name**.

.NET Adapter Wrapper

This section contains instructions for implementing and deploying the .NET adapter.

Note: In order to use .NET adapter wrapper, Microsoft Visual Studio 2008 and .NET Framework 3.5 must be installed on your computer.

Implementing the .NET Adapter

To implement the .NET adapter:

1. Extract the files from **op-adapter-dotnet.zip** to your hard drive.
2. Open the MSDEV project **op-adapter-dotnet\OpAdapterService.sln**.
3. Edit the default adapter configuration in the following file:

op-adapter-dotnet\AdapterService\config\default_adapter_config.xml

Note: The facet image URI should be in the format **"/images/X"**, where **X** is the image name in PNG format (without the extension).

4. Implement the Adapter interface. The skeleton implementation is placed in the **AdapterImpl.cs** file:
 - a. Implement **getConfigurationXmlPath()**

The default implementation returns the path of "op-adapter-dotnet\AdapterService\default_adapter_config.xml"
 - b. Implement **isBtoObjectSupported()**

This method should return a value of True if the Context object of a given type and ID is supported by the adapter.

Use a security token if needed.

c. Implement **getFacetData()**

This method should return the **facet_result** object. The default implementation shows a sample of how to do this.

Use a security token if needed.

d. Compile your MSDEV project and try to view **OpAdapterService.svc** in the browser.

Deployment Steps

1. To perform manual deployment use the Microsoft utility **WebDev.WebServer.exe** and follow the steps below.
 - a. Copy the adapter's binary files onto the machine on which the Enterprise Collaborationserver is deployed.
 - b. Run **op-adapter-dotnet\AdapterService\run_server.bat**
2. To perform deployment on the server, use the standard IIS deployment procedure.
3. For both types of deployment (1 and 2), configure the URL of the .NET adapter as described in Step 3 of ["Adapter Wrapper Deployment Steps" \(on page 23\)](#).

How to Add an Adapter

To add an adapter, follow the steps described in "Setup the Adapter" in the Enterprise Collaboration Installation and Configuration Guide.

Chapter 4

REST API Schemas

This section presents the schemas that must be implemented for received parameters and returned values in the REST API.

You, the adapter developer, should use these schemas to create the required XML code for the adapter.

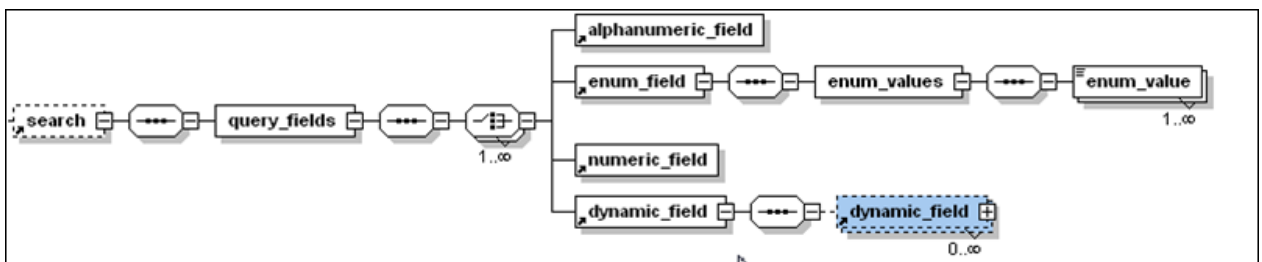
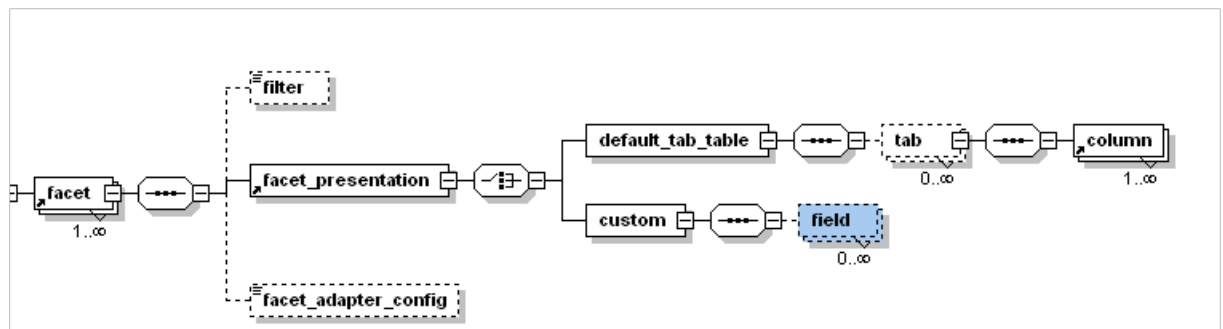
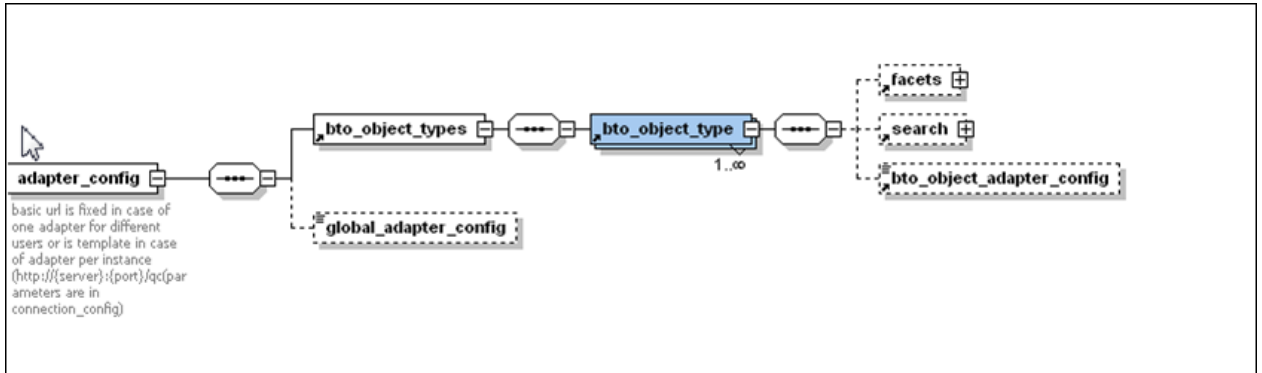
It contains the following schemas:

Adapter Configuration Schema	27
XSD Structure	27
Sample Code	30
Facet Result Schema	32
Sample Code	33
Context Object Schema	33
Sample Code	34
Search Result Schema	35
Sample Code	35

Adapter Configuration Schema

The `adapter_config.xsd` schema defines how the adapter configuration is set up.

The Adapter Configuration schema is presented in the following diagram:



XSD Structure

Element Name	Description		
adapter_config	The schema's root element. This element contains bto_object_types and global_adapter_config elements.		
	Required Attributes		
	<table border="1"> <tr> <td>name</td> <td>The adapter configuration name.</td> </tr> </table>	name	The adapter configuration name.
name	The adapter configuration name.		
	Optional Attributes		

Element Name	Description	
	instance_display_label	The name of the external application instance that will be shown in the search objects screen.
bto_object_types	Contains all Context object types supported by the adapter.	
global_adapter_config	Contains proprietary global adapter configuration information. Can contain the global information for adapter, i.e., application connection configuration.	
bto_object_type	Represents the type of a Context object. Contains facets, search and bto_object_adapter elements.	
	Required Attributes	
	name	The name of this Context object type. For example: "incident".
facets	Contains all the facets supported by the specific Context object type.	
bto_object_adapter_config	Contains proprietary adapter configuration for the specific Context object type. Can contain the source application API name for retrieving the data for this type.	
facet	Represents a facet of a specific Context object type. Contains filter, facet_presentation and facet_adapter_config definitions.	
	Required Attributes	
	name	The name of this facet. For example: "Incident details".
	image_url	URL to the icon that will represent the current facet. The URL is relative to basic adapter URL.
	Optional Attributes	
	Description	The description of this facet. For example: "All the incident details".
filter	An optional element for filtering the retrieved data. For example, the filter can contain a query for retrieving only open incidents. The query is written in the source application proprietary language.	

Element Name	Description	
facet_presentation	Contains definition for facet presentation. The presentation can be default (provided by Enterprise Collaboration) or custom (provided by the adapter). Currently there is only one default presentation (default_tab_table).	
facet_adapter_config	Contains proprietary adapter configuration for the specific facet. This element can contain the mapping for source application fields.	
field	Defines the one of the custom result fields.	
	Required Attributes	
	name	The name of the field.
	Optional Attributes	
	useInThumbnail	Whether or not fields should be used in the thumbnail presentation. Default is false.
default_tab_table	Defines the default tab table presentation provided by Enterprise Collaboration. It can contain the configuration per tab. Each tab contains the columns to present on it. In the current version, the presentation is simple table presentation only.	
	Required Attributes	
	name	The tab name.
column	Contains the column presentation definition in the table.	
	Required Attributes	
	name	The name of the column.
	Optional Attributes	
	title	Column title
	type	Column data type
	color	Column color
	size	Column size
	useInThumbnail	Whether or not fields should be used in the thumbnail presentation. Default is false.
search	Contains the configuration required for searching the objects from the current type.	

Element Name	Description
query_fields	Contains all the fields that can be used for searching the objects. All the fields have the following attributes: name and display_label and a mandatory flag
alphanumeric_field	Defines a field that contains alphanumeric values.
enum_field	Defines a field that contains static enumerated values
enum_values	Contains all enumerated values.
enum_value	Contains a single enumerated value.
numeric_field	Defines a field that contains numeric values.
dynamic_field	Defines a dynamic field that has values that depend on application instance, current user and values of previous dynamic fields. It contains the other dynamic_field elements that depend on the value of this field.

Sample Code

The following code shows an example of how the **adapter_config** schema can be used:

```
<adapter_config name="mock" instance_display_label="Mock Application">
  <bto_object_types>
    <bto_object_type name="mock">
      <facets>
        <facet name="Mock details" description="Mock details" image_url="/images/facet.png">
          <facet_presentation>
            <custom/>
          </facet_presentation>
        </facet>
      </facets>
      <search>
        <query_fields>
          <alphanumeric_field mandatory="true" name="filePath" display_label="File Path"/>
          <dynamic_field mandatory="false" name="dynamicField0" display_label="Dynamic Field 0">
          <dynamic_field mandatory="false" name="dynamicField1" display_label="Dynamic Field 1">
          <dynamic_field mandatory="false" name="dynamicField3" display_label="Dynamic Field 3"/>
          </dynamic_field>
          <dynamic_field mandatory="false" name="dynamicField2" display_label="Dynamic Field 2">
          </dynamic_field>
        </dynamic_field>
      </query_fields>
    </search>
  </bto_object_type>
</bto_object_types>
</adapter_config>
```

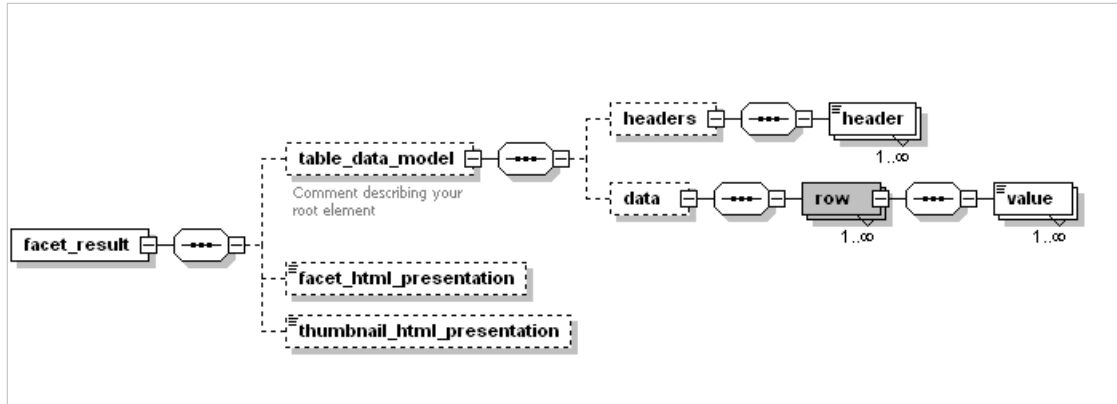
```
</bto_object_type>
</bto_object_types>
<global_adapter_config>
  <![CDATA[ <global_config >
    <connection_config>
      <config_property key="host" value="<host_name>"/>
      <config_property key="port" value="<port_number>"/>
    </connection_config>
  </global_config> ]]>
</global_adapter_config>
</adapter_config>
```

Facet Result Schema

The `facet_result.xsd` schema defines how the facet result returns a value.

The REST API structure returns this schema.

The Facet Result schema is presented in the following diagram:



Elements

Element	Description
facet_result	The schema's root element. This element contains table_data_model, facet_html_presentation and thumbnail_html_presentation elements. At least one of the table_data_model/facet_html_presentation elements should exist in the facet result.
table_data_model	Contains the facet data as a simple 2D table. This element is optional, and is used when a custom presentation is used. It contains headers and data elements.
facet_html_presentation	Contains the custom html presentation of facet. This element is optional, and is used when the custom presentation is used.
thumbnail_html_presentation	Contains the thumbnail custom html presentation of facet. This element is optional, and is used when the custom presentation is used.
headers	Contains all table headers.
data	Contains all table rows.
header	Represents the table column name.
row	Contains the table row values.
value	Contains the cell's value.

Sample Code

The following code shows an example of how the **facet_result** schema can be used:

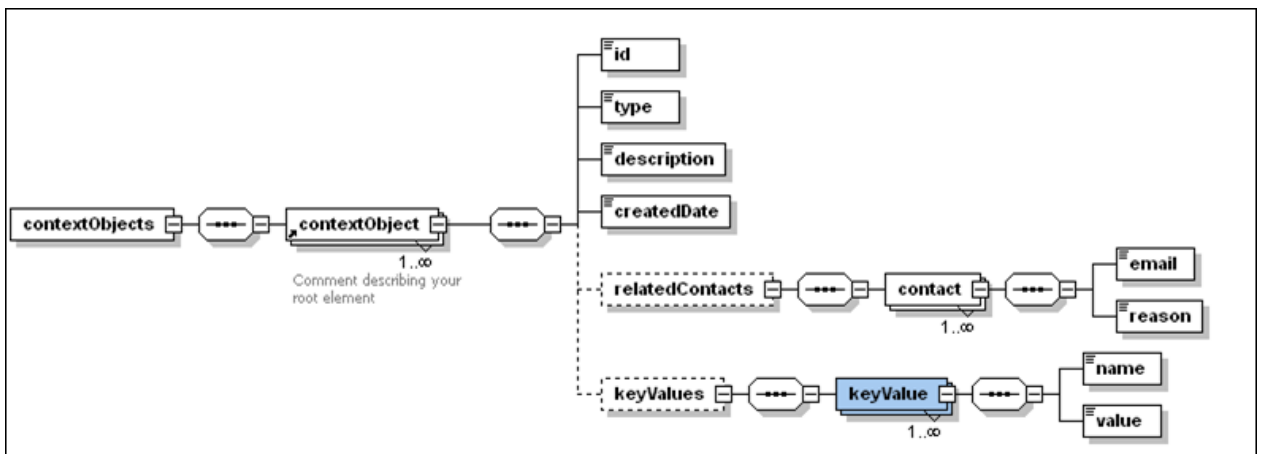
```
<facet_result>
  <table_data_model>
    <headers>
      <header>Title</header>
      <header>Description</header>
      <header>AdditionalHeader1</header>
      <header>AdditionalHeader2</header>
    </headers>
    <data>
      <row>
        <value>Title Value</value>
        <value>Description Value</value>
        <value>AdditionalHeader1 Value</value>
        <value>AdditionalHeader2 Value</value>
      </row>
    </data>
  </table_data_model>

  <facet_html_presentation><![CDATA[<html>facet_html_presentation</html>
]]>
  </facet_html_presentation>
  <thumbnail_html_presentation><![CDATA[<html>thumbnail_html_
presentation</html> ]]>
  </thumbnail_html_presentation>
</facet_result>
```

Context Object Schema

The **contextObject.xsd** schema defines the context object. This schema is returned by the REST API.

The Context Object schema is presented in the following diagram:



Sample Code

The following code shows an example of how the **contextObject** schema can be used:

```
<contextObjects>
  <contextObject>
    <id>Id1</id>
    <type>SampleType</type>
    <description>Object 1</description>
    <createdDate>2147483647</createdDate>
    <relatedContacts>
      <contact>
        <email>user1@org.com</email>
        <reason>owner</reason>
      </contact>

      <contact>
        <email> user2@org.com </email>
        <reason>assigned</reason>
      </contact>

      <contact>
        <email> user3@org.com</email>
        <reason>assigned</reason>
      </contact>
    </relatedContacts>

    <keyValues>
      <keyValue>
        <name>priority</name>
        <value>Urgent</value>
      </keyValue>
    </keyValues>
  </contextObject>
</contextObjects>
```

Search Result Schema

The **bto_object_search_result.xsd** defines the search result.

The Search Result schema is presented in the following diagram:



Sample Code

The following code shows an example of how the **bto_object_search_result** schema can be used:

```
<bto_object_search_result>
  <bto_object_descriptors>
    <bto_object_descriptor>
      <id>Id1</id>
      <type>IdType</type>
      <description>Id1 object description</description>
    </bto_object_descriptor>
    <bto_object_descriptor>
      <id>Id2</id>
      <type>IdType</type>
      <description>Id2 object description</description>
    </bto_object_descriptor>
  </bto_object_descriptors>
  <total>100</total>
</bto_object_search_result>
```

Chapter 5

Adapter JMX Methods

Enterprise Collaboration provides several useful methods to receive/update adapter configuration.

To access these methods:

1. Go to `<EC_Application_URL>/Diamond/jmx-console`.
2. Select **Diamond > Diamond adapter config JMX service**.

Chapter 6

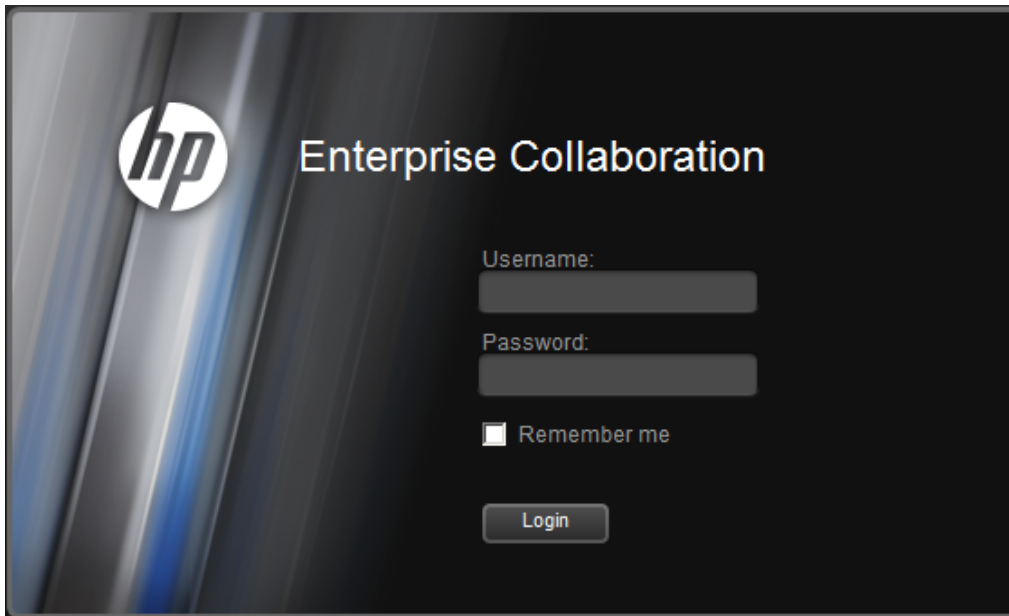
URL Launch

The Enterprise Collaboration URL Launch capability allows you to run Enterprise Collaboration directly via URLs called from within your application.

This chapter presents the URLs you should use to perform different operations on the Enterprise Collaboration conversations. It contains the following sections:

Authentication	38
Display Inbox URL	38
Search Conversations by Context Objects URL	39
Context Objects XML Structure	40
URL Attachments	41
Sample URL Attachments XML	41
Search Results	41

After calling the URL, the Enterprise Collaboration Login screen opens:



After logging in, the relevant page (as called by the URL) opens.

Authentication

In order to perform authentication, you need to pass the LWSSO cookie containing the user credentials into the request header.

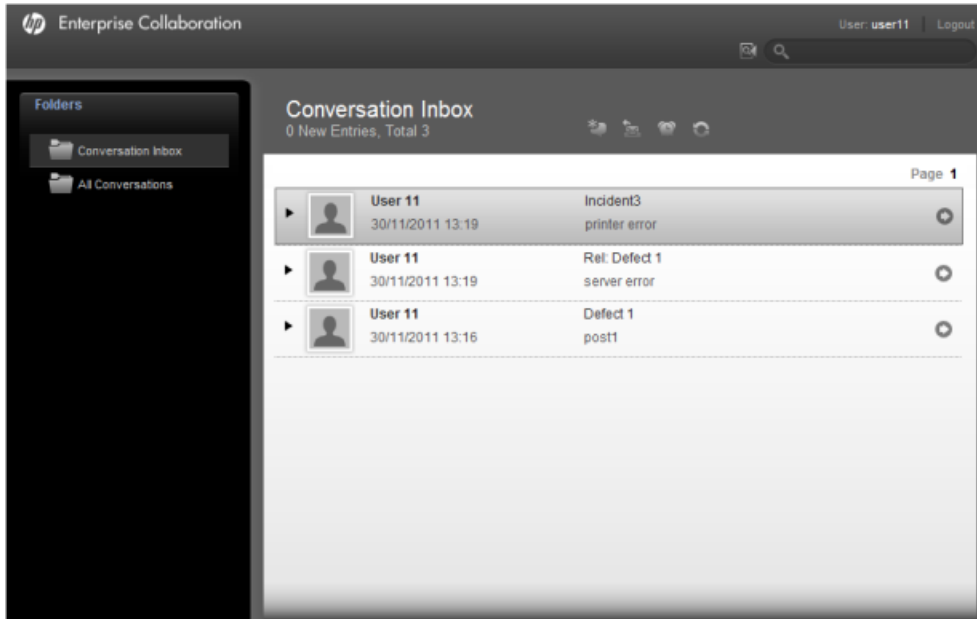
Alternately, you can pass the user token containing the user credentials into the URL. However, this method is not recommended.

Display Inbox URL

To display all the conversations in the Enterprise Collaboration Conversation Inbox, go to the following URL:

<EC_Application_URL>/diamond

A sample Conversation Inbox is shown below:



Search Conversations by Context Objects URL

You can launch Enterprise Collaboration to display conversations containing specific context objects by calling **<EC_Application_URL>/diamond/DiamondByURLServlet**

Syntax

```
<EC_Application_URL>/diamond/DiamondByURLServlet {?subject=<subject>}  
{&object=<objectid>}&SearchbyObjectId=<objectid>  
{&SearchbyObjectId=<objectid>}  
{searchAtLeastOne}
```

Parameters

All parameters should be URL encoded:

1. **subject**: The subject assigned to new conversations that the user creates after the URL launch.
2. **objects**: context objects in xml format (as shown in the example "[Search Conversations by Context Objects URL](#)"). These objects will be attached to any new conversation that the user creates after the URL launch.
3. **[urlAttachments]**: Attachments in xml format (as shown in the example "[Sample URL Attachments XML](#)"). These URL attachments will be attached to any new conversation that the user creates after the URL launch.
4. **searchByObjectId**: id of context object. Enterprise Collaboration search conversations that contain this object. In order to perform search with several ids, you should specify this parameter for each id.

For example:

```
<EC_Application_URL>/diamond/DiamondByURLServlet  
?subject=Defect1&searchByObjectId=id1&searchByObjectId=id2
```

5. `searchAtLeastOne`: Optional parameter. By default, Enterprise Collaboration search conversations that contain all the objects specified in `searchByObjectId` parameters (and condition). To search conversations that contain at least one object, add `&searchAtLeastOne=true` to the URL.

Context Objects XML Structure

A sample Context Objects XML structure is presented below:

```
<contextObjects>
  <contextObject>
    <id>IM100ds02</id>
    <type>defect</type>
    <description>Defect object 1</description>
    <createdDate>2147483647</createdDate>
    <relatedContacts>
      <contact>
        <email>developer1@company.com</email>
        <reason>owner</reason>
      </contact>
      <contact>
        <email>developer2@company.com</email>
        <reason>assigned</reason>
      </contact>
    </relatedContacts>
    <keyValues>
      <keyValue>
        <name>priority</name>
        <value>Urgent</value>
      </keyValue>
    </keyValues>
  </contextObject>
  <contextObject>
    ...
  </contextObject>
</contextObjects>
```

Note: The `relatedContacts` and `keyValues` sections are optional.

URL Attachments

Internal or external applications can pass attachments containing URLs to an Enterprise Collaboration conversation. For example, the Executive Scorecard application can pass a URL containing details of KPI information to a specific conversation in EC. This URL is added as an attachment to the conversation.

The application can provide the URL attachment(s) in an XML string with a single element `<urlAttachments>`, containing one or more `<urlAttachment>` elements (see the example in ["Sample URL Attachments XML" \(on page 41\)](#).)

EC checks the URL for validity. If a URL is invalid, it is not attached to the conversation.

Note: Passing an invalid URL attachment via URL launch does not produce a warning/error message to the user, but it will be logged in the log file.

Sample URL Attachments XML

A sample URL Attachments XML structure is presented below:

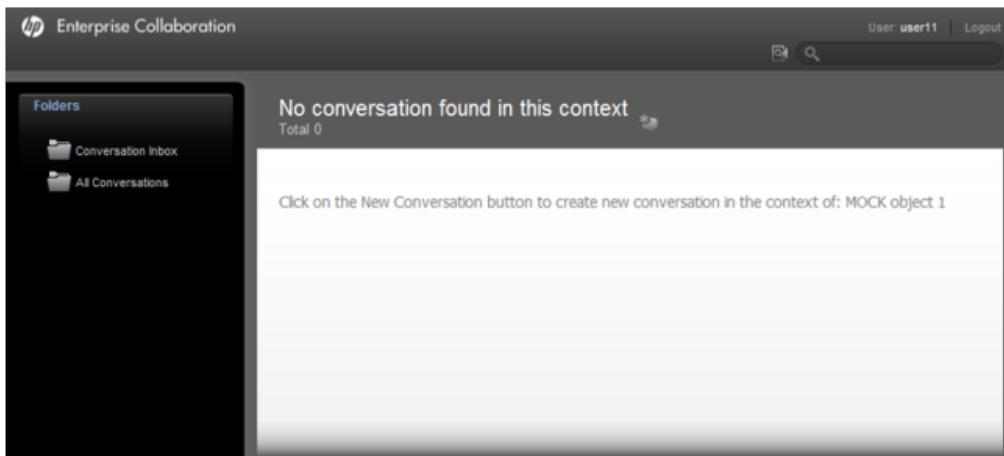
```
<urlAttachments>
  <urlAttachment>
    <url>http://www.hp.com/</url>
    <description>HP home</description>
  </urlAttachment>
</urlAttachments>
```

Search Results

This section presents the possible results of a Search Conversations operation.

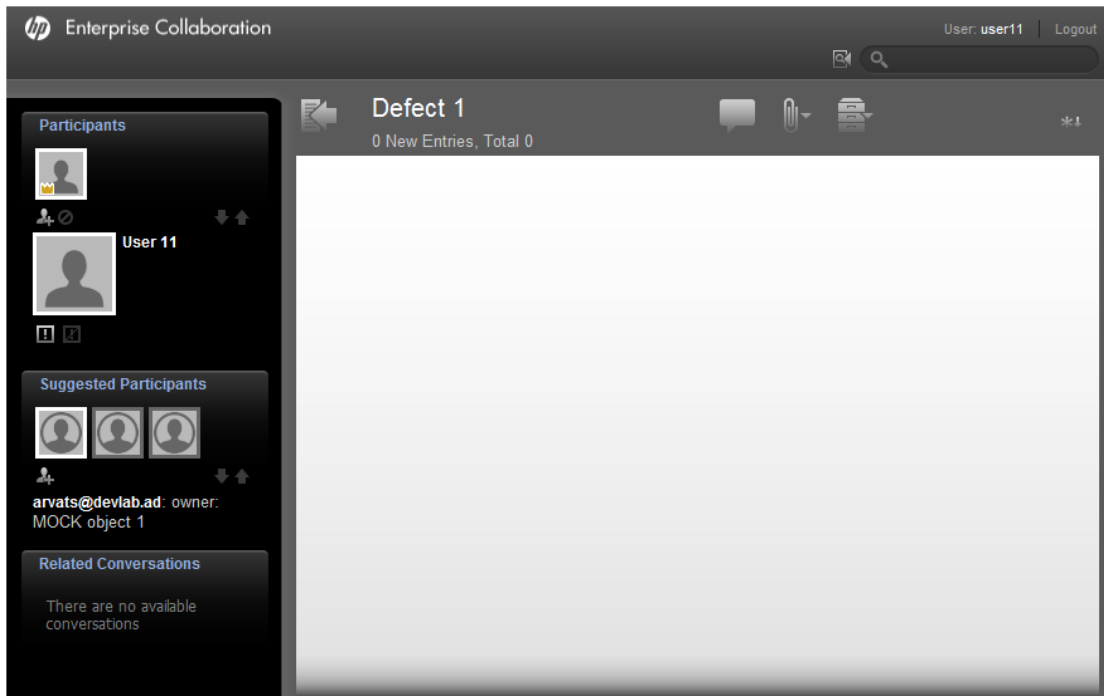
1. No conversation contains the specified object ids

If there are no conversations that meet the search criteria, the following message is shown:



2. One conversation contains the specified object ids

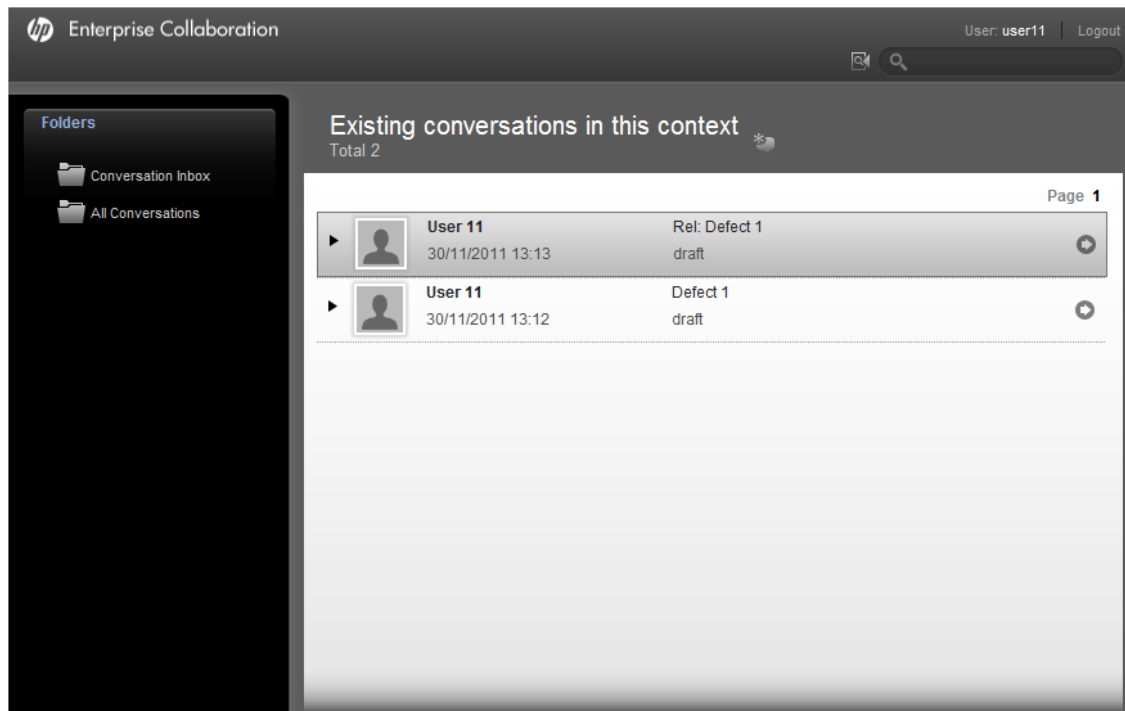
If there is one conversation that meets the search criteria, Enterprise Collaboration automatically opens the matching conversation as shown in the example below:



Select the **Back to List** icon  to show your Inbox.

3. Several conversations contain the specified object ids

If there is more than one conversation that meets the search criteria, Enterprise Collaboration displays a list of all conversations that match the search criteria. For example, if you search for the term "Defect" the following items are shown:



To create a new conversation from the Search Results screen:

- Click the **New Conversation** icon .

The new conversation is assigned with the subjects and objects specified in the URL Launch parameters.

- Select the Inbox folder (**Conversation Inbox**) to view all conversations in your inbox.

Note: New conversation created in this mode are not assigned with the specified subject and objects.

For details on the Enterprise Collaboration User Interface, see the HP Enterprise Collaboration Concepts Guide.

Chapter 7

Compact Mode

When Enterprise Collaboration is embedded into another application and does not use the entire screen size, it is recommended to launch it in Compact mode. In compact mode, the user interface is rendered so that it does not use the entire screen size.

In this way, the user get the same functionality in less space, and can have Enterprise Collaboration on top of another application.

To launch Enterprise Collaboration in Compact mode:

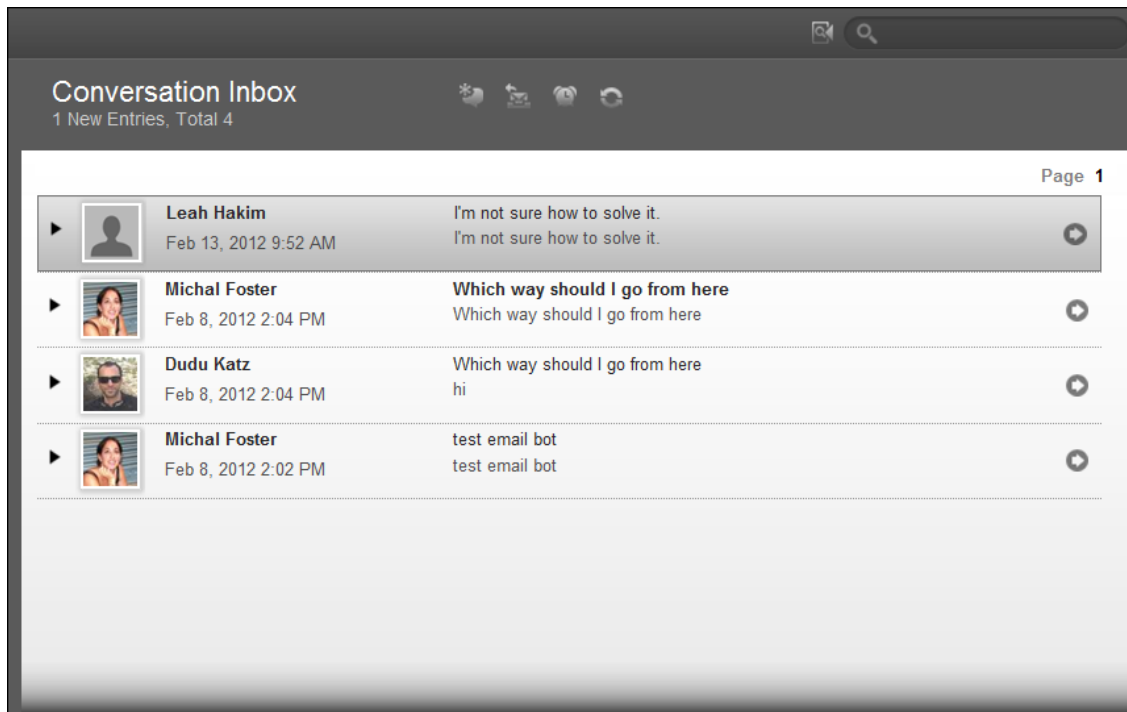
- Add **&compactMode=true** to the URL.

Compact Mode Functionality

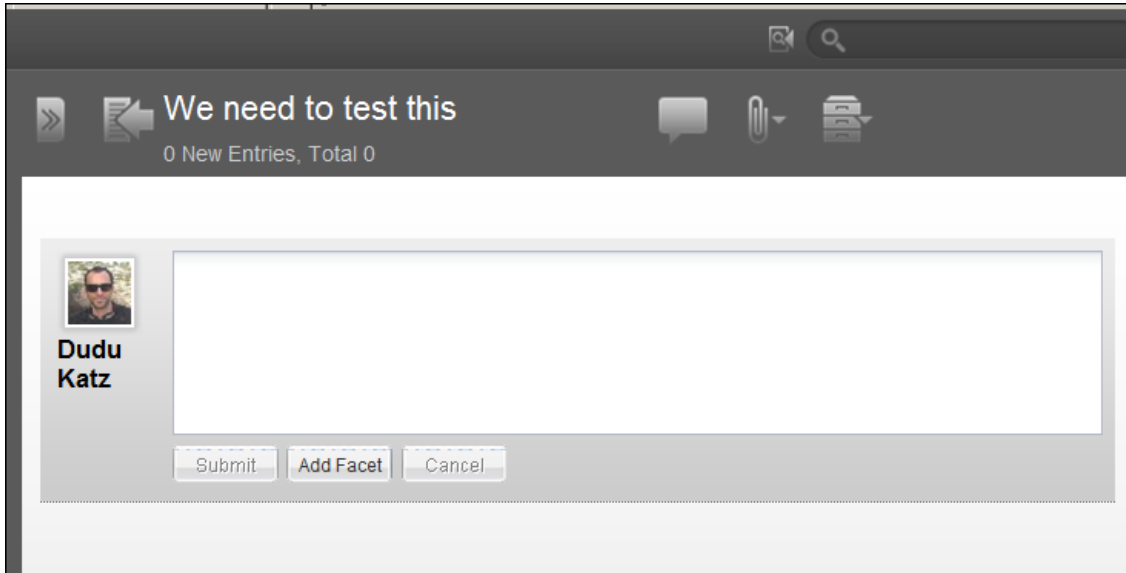
In Compact mode, the following elements are not visible in the Conversation window, as their functionality is provided by the host application:


- Enterprise Collaboration logout
- User name
- Logout link

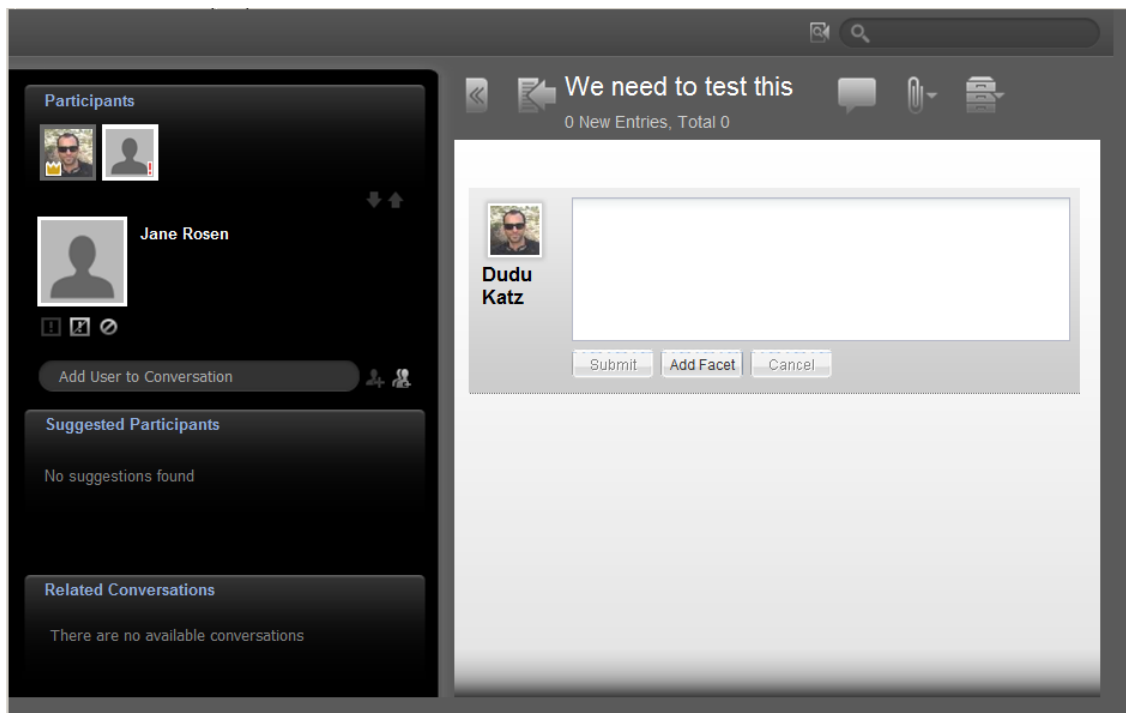
In addition, the folders panel is collapsed and you cannot see the User Inbox in URL launch flow .



When you enter a conversation, the left panel (containing participants, suggested and related conversations) is collapsed.



- Click  to expand the left panel:



Chapter 8

JavaScript Integration

A more efficient and less limited method of running Enterprise Collaboration is using the JavaScript integration.

Using the JavaScript integration, you can search for objects in conversations by invoking a JavaScript function on a live instance of Enterprise Collaboration instead of using URL Launch, that loads all conversations for each object.

In addition, using the JavaScript integration, there is no limit to parameter size (in the GET method). However, URLs containing over 2,000 characters are generally not supported in most popular web browsers.

To pass large context objects you can use one of the following methods:

- Use the “POST” method (instead of “GET”)
- Invoke JavaScript to perform the search. The advantage of using JavaScript is that there is no size limit on the parameters.

Implementing JavaScript in Enterprise Collaboration

In order to interact with Enterprise Collaboration you first need to launch it with an authenticated user (once only for each user).

Use the following URL to launch Enterprise Collaboration and enable the JavaScript integration:

<EC_Application_URL>/diamond/?action=getConversationsListByIds

After Enterprise Collaboration is loaded, you can perform a search by invoking the relevant JavaScript function as described below.

Search According to One Object Id Call

To search according to one object id call, use the following JavaScript function:

searchConversationsByObjectId(conversationSubject, objectsContent, urlAttachments, searchByObjectId)

Parameters

1. **subject**: The subject assigned to new conversations that the user creates after the URL launch.
2. **objects**: context objects in xml format (as shown in the example "[JavaScript Integration](#)"). These objects will be attached to any new conversation that the user creates after the URL launch.
3. **urlAttachments**: Attachments in xml format (as shown in the example "[Sample URL Attachments XML](#)"). These URL attachments will be attached to any new conversation that the user creates after the URL launch.

4. `searchByObjectId`: id of context object. Enterprise Collaboration search conversations that contain this object. In order to perform search with several ids, you should specify this parameter for each id.

For example:

```
<EC_Application_URL>/diamond/DiamondByURLServlet  
?subject=Defect1&searchByObjectId=id1&searchByObjectId=id2
```

5. `searchAtLeastOne`: Optional parameter. By default, Enterprise Collaboration search conversations that contain all the objects specified in `searchByObjectId` parameters (and condition). To search conversations that contain at least one object, add `&searchAtLeastOne=true` to the URL.

Note: The function parameters are identical to the URL Launch parameters (see ["URL Launch" \(on page 37\)](#)), except that URL encoding is not used here.

Search According to Several Object Id Calls

To search according to several ids call, use the following JavaScript function:

```
searchConversationsByObjects(conversationSubject, objectsContent,  
searchByObjectIds)
```

Parameters

1. `conversationSubject` – String
2. `objectsContent` – String
3. `searchByObjectIds` – String[]

The function parameters are identical to the URL Launch parameters (see ["URL Launch" \(on page 37\)](#)), except that URL encoding is not used here.

Check Enterprise CollaborationStatus

The element `hpEcStatus` contains the current status of Enterprise Collaboration. During launching, `hpEcStatus` is "Loading".

After loading has completed, `hpEcStatus` is changed to "Ready".

To check the current status, use the following code:

```
Document.getElementById("hpEcStatus").getAttribute("status")
```

.NET Example for JavaScript Integration

The following JavaScript .NET example demonstrates how to launch Enterprise Collaboration and invoke JavaScript from .NET code using the WebBrowser component.

```
namespace ECTest
{
    public partial class Form1 : Form
    {
        #region Constants

        private const string JAVA_SCRIPT_FUNCTION_NAME =
            "searchConversationsByObjectId";
        private const string EC_STATUS_ELEMENT_ID = "hpEcStatus";
        private const string EC_STATUS_ATTRIBUTE_NAME = "Status";
        private const string EC_READY_STATUS_VALUE = "Ready";
        string objects =
            "<contextObjects><contextObject>"
+
            "  <id>IM100ds02</id>"
+
            "  <type>defect</type>"
+
            "  <description>object 1</description>"
+
            "  <createdDate>2147483647</createdDat>"
+
            "    <relatedContacts>"
+
            "      <contact>"
+
            "        <email>user1@devlab.ad</email>"
+
            "        <reason>owner</reason>"
+
            "      </contact>"
+
            "      <contact>"
+
            "        <email>user2@devlab.ad</email>"
+
            "        <reason>assigned</reason>"
+
            "      </contact>"
+
            "      <contact>"
+
            "        <email>user3@devlab.ad</email>"
            "
```



```
+      " <reason>assigned</reason>                                "
+      " </contact>                                              "
+      " </relatedContacts>                                       "
+      "      <keyValues>                                          "
+      " <keyValue>                                              "
+      " <name>priority</name>                                       "
+      " <value>Urgent</value>                                       "
+      " </keyValue>                                              "
+      " </keyValues>                                           "
+      "</contextObject>                                          "
+      "</contextObjects>                                         ";

String urlDiamond = "http://ECSite:8080/diamond/";

#endregion

private bool ecReady;

public Form1 ()
{
    InitializeComponent();

    webBrowser1.Navigate(new Uri(urlDiamond));
}

private void webBrowser1_DocumentCompleted(object sender,
WebBrowserDocumentCompletedEventArgs
e)
{
    toolStripTextBoxURL.Text = e.Url.AbsoluteUri;
}

private void toolStripButton1_Click(object sender,
EventArgs e)
{
    HTMLElement ecStatus = webBrowser1.Document.
        GetElementById(EC_STATUS_
```

```
ELEMENT_ID);
    if (ecStatus != null)
    {
        ecReady = ecStatus.GetAttribute(EC_STATUS_
ATTRIBUTE_NAME).
            Equals(EC_READY_STATUS_VALUE,
                StringComparison.CurrentCultureIgnoreCas-
e);
        if (ecReady)
        {
            webBrowser1.Document.InvokeScript
                (JAVA_SCRIPT_FUNCTION_NAME, new
object[]
                { "testSubject", objects,
                "ef7aaea2-d0ed-4512-a726-334fb4f446aa"
            });
        }
        else
        {
            MessageBox.Show("Page not initialized");
        }
        else
        {
            MessageBox.Show("Page not loaded");
        }
    }
}
```

Glossary

Context Object

An object that represents a specific entity of an application.

facet

A presentation of a Context object.

