Web Services Interface

This guide describes the Web Services interface within ExtraView. The intended audience is experienced developers who wish to use the Web Services interface to integrate ExtraView to remote external applications that also support a Web Services interface.

ExtraView provides a standardized set of methods to integrate with other applications using a service-orientated architecture (SOA). This complements the web-orientated architecture interface (WOA) that is also implemented within ExtraView and described in the complementary ExtraView Command Line Interface and Application Programming Interface guide. The interface is cross platform and can be accessed from any development platform, including Java and .Net.

A service-oriented architecture is defined as a group of services, which communicate with each other. The process of communication involves either simple data passing or it involves two or more services coordinating some activity. Some means of connecting services to each other is needed.

SOA applications are built out of software services. Services are intrinsically unassociated units of functionality, which have no calls to each other embedded in them. Within ExtraView they map to atomic functions to perform specific actions. Broadly, SOAs implement functionalities most humans would recognize as a service, such as filling out an online application for an account, viewing an online bank statement, inserting an item with a database or running a report. Instead of services embedding calls to each other in their source code, protocols are defined which describe how one or more services can talk to each other. This architecture then relies on your business process expert to link and sequence services to meet your business system requirement.

ExtraView's implementation of web services utilizes a service-orientated architecture to provide a full set of integration points between ExtraView and the consumer of its web services.

Documentation Examples

The examples in this documentation were prepared using Java as the foundation. There is no certainty that the helper methods used are available in other languages or platforms. It is expected that the experienced developer will recognize this and be able to use the techniques described and adapt them for their own use.

Downloadable PDF

The Web Services Interface Guide is downloadable as a single PDF by clicking here. You will need the Adobe Acrobat Reader to view this.

Installation & Configuration

The ExtraView Web Service module is packaged as a war file. The configuration file is held inside of this war file, and must be edited for your environment. If you do not have the download, ExtraView Support will give you the location where the file resides. Download this before starting the installation. The following instructions utilize Apache Tomcat as the application server and Apache as your web server. If

you are using different servers, it is assumed you have sufficient knowledge to modify the instructions yourself. To install the Web Service:

- Start by ensuring your application server is running
- The war file you download will have a name similar to evjnnnn-WS.war, where the nnnn is the version and build number of the file. In all the following instructions, replace the nnnn with the actual version and build number of the file
- Rename the evjnnnn-WS.war file to evj-WS.war. You may use a different file name as long as its name does not override an existing file, but once again modify the following instructions to use the name you supply
- Copy the war file to the web application directory of you application server. Normally this is the directory named webapps within your application server
- The application server will expand the Web Service war file, extracting it into a directory named ./evj-WS
- Using the editor of your choice, edit the Web Service configuration file named ./evj-WS/WEB-INF/classes/ev_ws.properties. The contents of the configuration file will be similar to the following:

```
# (REQUIRED) The URL for the ExtraView API
API_URI=http://yourserver.yourdomain.com/evj/ExtraView/ev_api.action

# (OPTIONAL) The ExtraView user account and password used for PING only
EV_WS_USER=admin
EV_WS_PASSWORD=password

# (OPTIONAL) Used for setting up full RAMPART secure message access to the web
service SECURE_ID=service
SECURE_PASSWORD=apache

# (OPTIONAL) The internal expected delimiter (must match the users behavior
setting)
DELIMITER=:
```

- If your application server does not expand the war file automatically, you will need to open the war file, edit the configuration file, and re-create the war file before copying it to your application server
- You must change the API_URI entry to match your ExtraView server location. Replace the http://yourserver.yourdomain.com text with the actual location of your ExtraView server
- Alter the EV_WS_USER and the EV_WS_PASSWORD entries to match those of a valid ExtraView user
- For the initial installation you may ignore the SECURE_ID and the SECURE_PASSWORD entries
- Save the file
- You must update the web server configuration file to include the web services and its associated WSDL file. Edit the httpd.conf file and add the following lines:

  JkMount /evj-WS/services/* tomcat1
  JkMount /evj-WS/wsdl/* tomcat1

  Alternatively, if you are using an ev.conf or extraview.conf file, you add these mounts into that file. Make sure you save the file after editing

- Restart your application server
- Validate the installation works with the getPing service. The URL to call this will be:

http://SERVER_NAME/evj-WS/services/EVSystemService/getPing

A successful response will look similar to:

<ns:getPingResponse>
  <ns:return>
    Reply: status=SUCCESS: EV-EXTRAVIEW ALIVE, WS-EV-PROPERTIES-OK, EV-REV-UNKNOWN-OK time=94ms   </ns:return>
</ns:getPingResponse>

- The WSDL files for your web services are automatically generated as part of the installation. Their URL's are:

  http://yourserver.yourdomain.com/evj-WS/services/EVSystemService?wsdl
  http://yourserver.yourdomain.com/evj-WS/services/EVUserService?wsdl
  http://yourserver.yourdomain.com/evj-WS/services/EVItemService?wsdl
  http://yourserver.yourdomain.com/evj-WS/services/EVQueryService?wsdl

Web Services Functions

The functions are broken into four basic groups, each with a list of calls that are appropriate to the group. This guide provides details of each call, along with details of the parameters that they use, and the results returned from the service. An example of each call is provided within the guide.

User Calls

```
insertUser
updateUser
setCurrentUserRole
getUserByEmail
getUserById
getUserFields
getUserRoles
disableUser
getRolesByUser
insertUserRole
```

System Calls

```
getPing
getVersion
getFields
getAppDefault
getHeartbeat
getFieldTitle
setWorkingProjectArea
getAllowedFieldValues
executeCustomCode
```

Report Calls

```
executeReport
executeItemSearch
getFieldList
executeUserSearch
```

Item Calls

```
getItemFieldList
deleteItem
getChangedItems
getItemAudits
insertItem
updateItem
getItem
itemExists
getItemAttachments
```

addItemAttachment

- Add new comment

The addItemAttachment service attaches a file to an existing item in the ExtraView database.

Input

Class	Name	Type	Required	Details
AddItemAttachmentRequest	userId	String	Yes	The caller's user ID
AddItemAttachmentRequest	password	String	Yes	The caller's password
AddItemAttachmentRequest	attachment	Object		The data handler object as defined by the soap product your using
AddItemAttachmentRequest	itemId	int	?	The int item id. This field is optional if itemIdStr is given, else it is required. It is recommended that you use itemIdStr
AddItemAttachmentRequest	itemIdStr	String	?	The String item Id. This field

				is optional if itemId is used. It is recommended that you use itemIdStr
AddItemAttachmentRequest	attachmentName	String	No	
AddItemAttachmentRequest	attachmentDesc	String	No	List of filters for the history
AddItemAttachmentRequest	charset	String	No	
AddItemAttachmentRequest	contentType	String	No	
AddItemAttachmentRequest	fieldName	String	No	
AddItemAttachmentRequest	attachmentDesc	String	No	
AddItemAttachmentRequest	rowNumber	String	No	

Output

Class	Name	Type	Required	Details
AddItemAttachmentResponse	success	boolean	Yes	True is succeeded False if failed
AddItemAttachmentResponse	returnCode	String	No	See Appendix for details
AddItemAttachmentResponse	returnMessage	String	No	Human readable message

Example

```
public static void testAddItemAttachment(EVItemServiceStub stub) {
```

```
    try {
        // generate a temp file
        File tempFile = File.createTempFile("ws_test",".txt");
        FileOutputStream fop = new FileOutputStream(tempFile);
        fop.write("!!TEST FILE!!nWS Java Tester built this file!".getBytes());
        fop.flush();
        fop.close();

        // next setup the data source
        DataInputStream dis = new DataInputStream(new FileInputStream(tempFile));
        BufferedReader br = new BufferedReader(new InputStreamReader(dis));

        List bFileList = new ArrayList();

        try {
            byte b;
            while( (b=dis.readByte()) != -1){
                bFileList.add(b);
            }
        } catch (Exception e) {}

        // and now we can send the file
        AddItemAttachmentDocument reqEnvelope =
AddItemAttachmentDocument.Factory.newInstance();
        AddItemAttachmentRequest request =
reqEnvelope.addNewAddItemAttachment().addNewRequest();

        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setItemId(CREATED_Item_ID);
        byte[] results = new byte[bFileList.size()];
        for (int i=0;i
            results[i] = bFileList.get(i);
        }
        request.setAttachment(results);
        request.setAttachmentName(tempFile.getName());
        request.setCharset("US-ASCII");
        request.setContentType("text/plain");
        request.setAttachmentDesc("this is a test file!");

        AddItemAttachmentResponseDocument resEnvelope =
stub.addItemAttachment(reqEnvelope);
        AddItemAttachmentResponse response =
resEnvelope.getAddItemAttachmentResponse().getReturn();

        if (response.getSuccess()) {
            System.out.println("success: [" +
                    response.getReturnCode() + "] : " +
                    response.getReturnMessage());
            System.out.println("Attachment created for issue " + request.getItemId());
        } else {
            System.out.println("failure: [" +
                    response.getReturnCode() + "] : " +
                    response.getReturnMessage());
        }
         tempFile.delete();
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

deleteItem

This action allows you to delete an existing record within ExtraView's database. Note that you must have permission to delete records before you can execute this action. The security key that controls this is named PR_RESOLUTION.DELETE_BUTTON.

Input

Class	Name	Type	Required	Details
DeleteItemRequest	userId	String	Yes	The callers user name
DeleteItemRequest	password	String	Yes	The callers password
DeleteItemRequest	itemId	int	?	The int item id. This field is optional if itemIdStr is given, else it is required. It is recommended that you use itemIdStr
DeleteItemRequest	itemIdStr	String	?	The String item Id. This field is optional if itemId is used. It is recommended that you use itemIdStr

Output

Class	Name	Type	Required	Details
DeleteItemResponse	success	boolean	Yes	True is succeeded False if failed
DeleteItemResponse	returnCode	String	No	See Appendix for details
DeleteItemResponse	returnMessage	String	No	Human readable message

Example

```
public static void testDeleteItem(EVItemServiceStub stub) {
    try {
        DeleteItemDocument reqEnvelope = DeleteItemDocument.Factory.newInstance();
        DeleteItemRequest request = reqEnvelope.addNewDeleteItem().addNewRequest();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setItemId(CREATED_Item_ID);
        DeleteItemResponseDocument resEnvelope = stub.deleteItem(reqEnvelope);
        DeleteItemResponse response = resEnvelope.getDeleteItemResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

disableUser

This action disables a user account within ExtraView. Note that there is no facility to delete a user account. This is to keep referential integrity within the database. Users that you want to disable will have created and/or updated issues and their details are required historically in order to be able to display their information.

Input

Class	Name	Type	Required	Details
RemoveUserRequest	userId	String	Yes	The callers user name
RemoveUserRequest	password	String	Yes	The callers password
RemoveUserRequest	deleteUserId	String	Yes	The user id you want to delete

Output

Class	Name	Type	Required	Details

RemoveUserResponse	success	boolean	Yes	True is succeeded False if failed
RemoveUserResponse	returnCode	String	No	See Appendix for details
RemoveUserResponse	returnMessage	String	No	Human readable message

Example

```
public static void testDisableUser(EVUserServiceStub stub) {
    try {
        DisableUserDocument reqEnvelope = DisableUserDocument.Factory.newInstance();
        DisableUserRequest request = reqEnvelope.addNewDisableUser().addNewRequest();
        request.setRemoveUserId(ServiceClientHelper.generatedUserId);
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        DisableUserResponseDocument resEnvelope = stub.disableUser(reqEnvelope);
        DisableUserResponse response = resEnvelope.getDisableUserResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

executeCustomCode

This call invokes the CLI user exit in the UserCustom java class. This implies that there will be additional code written within the UserCustom class to support the call. The power of this is that you can extend the API with your own commands written for your own purposes, to complement the commands and calls documented in this guide.

Input

Class	Name	Type	Required	Details
ExecuteCustomCodeRequest	userId	String	Yes	The callers user name

ExecuteCustomCodeRequest	password	String	Yes	The callers password
ExecuteCustomCodeRequest	parameters	Array[]	Yes	An array of name, value pairs.
FieldMetaData	name	String	Yes	The field name
FieldMetaData	fixedValue	String	Yes	The field value
FieldMetaData	value	String	No	The field value

Output

Class	Name	Type	Required	Details
ExecuteCustomCodeResponse	success	boolean	Yes	True is succeeded False if failed
ExecuteCustomCodeResponse	returnCode	String	No	See Appendix for details
ExecuteCustomCodeResponse	returnMessage	String	No	Human readable message

Example

```
public static void testExecuteCustomCode(EVSystemServiceStub stub) {
    try {
        ExecuteCustomCodeDocument reqEnvelope =
ExecuteCustomCodeDocument.Factory.newInstance();
        ExecuteCustomCodeRequest request =
reqEnvelope.addNewExecuteCustomCode().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        FieldMetaData[] parameters = new FieldMetaData[1];
        parameters[0] = request.addNewParameters();
        parameters[0].setName("foo");
        parameters[0].setValue("bar");
        request.setParametersArray(parameters);
        ExecuteCustomCodeResponseDocument resEnvelope =
stub.executeCustomCode(reqEnvelope);
```

```
        ExecuteCustomCodeResponse response =
           resEnvelope.getExecuteCustomCodeResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

executeItemSearch

This API call allows you to search the ExtraView database and to return a set of records that match the search criteria. This function is equivalent to the search capability within the browser version of ExtraView. It is extremely powerful as multiple search filters can be set on different fields. For example, it is straightforward to set up a search that responds to a query such as “tell me all the open items against a specific module within a specific product that contain a specific keyword.

Input

Class	Name	Type	Required	Details
ExecuteItemSearchRequest	userId	String	Yes	The callers user name
ExecuteItemSearchRequest	password	String	Yes	The callers password
ExecuteItemSearchRequest	status	String	No	The status of the items being collected
ExecuteItemSearchRequest	pageLength	Integer	No	The page length
ExecuteItemSearchRequest	persistHandle	String	No	A random string to persist the data
ExecuteItemSearchRequest	recordStart	Integer	No	The starting index of records returned

Output

Class	Name	Type	Required	Details
ExecuteItemSearchResponse	success	boolean	Yes	True is succeeded False if failed
ExecuteItemSearchResponse	returnCode	String	No	See Appendix for details
ExecuteItemSearchResponse	returnMessage	String	No	Human readable message
ExecuteItemSearchResponse	xmlResults	String	No	The search results in XML format
ExecuteItemSearchResponse	itemRecords	Array[]	Yes	An array of ItemRecord objects
ItemRecord	numberOfItemRecordFields	int	Yes	The number of field records in the child array
ItemRecord	itemRecordFields	Array[]	Yes	An array of ItemRecordField objects
ItemRecordField	fieldId	String	Yes	The id of the field
ItemRecordField	fieldTitle	String	Yes	The title of the field
ItemRecordField	fieldValue	String	Yes	The value of the field

ItemRecordField	row	int	Yes	The repeating row value

Example

```
public static void testExecuteIssueSearch(EVReportServiceStub stub) {
    try {
        ExecuteItemSearchDocument reqEnvelope =
ExecuteItemSearchDocument.Factory.newInstance();
        ExecuteItemSearchRequest request =
            reqEnvelope.addNewExecuteItemSearch().addNewRequest();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setPageLength(100);
        request.setRecordStart(1);
        request.setRecordCount(120);
        request.setPersistHandle(ServiceClientHelper.randStr);
        request.setReturnRawXML(true);
        request.setStatus("");
        ExecuteItemSearchResponseDocument resEnvelope =
stub.executeItemSearch(reqEnvelope);
        ExecuteItemSearchResponse response =
            resEnvelope.getExecuteItemSearchResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

executeReport

This function runs an existing report, using its *report_id*. This is obtained from the *get_reports* function.

Input

Class	Name	Type	Required	Details
ExecuteReportRequest	userId	String	Yes	The caller's user name
ExecuteReportRequest	password	String	Yes	The caller's password

Output

Class	Name	Type	Required	Details
ExecuteReportResponse	success	boolean	Yes	True is succeeded False if failed
ExecuteReportResponse	returnCode	String	No	See Appendix for details
ExecuteReportResponse	returnMessage	String	No	Human readable message
ExecuteReportResponse	xmlResults	String	No	The search results in XML format
ExecuteReportResponse	itemRecords	Array[]	Yes	An array of ItemRecord objects
ItemRecord	numberOfItemRecordFields	int	Yes	The number of field records in the child array
ItemRecord	itemRecordFields	Array[]	Yes	An array of ItemRecordField objects
ItemRecordField	fieldId	String	Yes	The id of the field
ItemRecordField	fieldTitle	String	Yes	The title of the field
ItemRecordField	fieldValue	String	Yes	The value of the field
ItemRecordField	row	int	Yes	The repeating row value

Example

```
public static void testExecuteReport(EVReportServiceStub stub) {
    try {
        ExecuteReportDocument reqEnvelope = ExecuteReportDocument.Factory.newInstance();
        ExecuteReportRequest request =
reqEnvelope.addNewExecuteReport().addNewRequest();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setPageLength(100);
        request.setRecordStart(1);
        request.setRecordCount(120);
        request.setPersistHandle(ServiceClientHelper.randStr);
        request.setReportId(ServiceClientHelper.STANDARD_REPORT);
        ExecuteReportResponseDocument resEnvelope = stub.executeReport(reqEnvelope);
        ExecuteReportResponse response =
resEnvelope.getExecuteReportResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

executeReportRaw

This function runs an existing report, using its *report_id*. This is obtained from the *get_reports* function and returns raw XML results.

Input

Class	Name	Type	Required	Details
ExecuteReportRawRequest	userId	String	Yes	The caller's user name
ExecuteReportRawRequest	password	String	Yes	The caller's password
ExecuteReportRawRequest	pageLength	Integer	Yes	The page length
ExecuteReportRawRequest	persistHandle	String	No	A random string to persist the data

ExecuteReportRawRequest	recordStart	Integer	No	The starting index of records returned
ExecuteReportRawRequest	recordCount	Integer	No	The count of records being fetched

Output

Class	Name	Type	Required	Details
ExecuteReportRawResponse	success	boolean	Yes	True is succeeded False if failed
ExecuteReportRawResponse	returnCode	String	No	See Appendix for details
ExecuteReportRawResponse	returnMessage	String	No	Human readable message
ExecuteReportRawResponse	xmlResults	String	No	The search results in XML format

Example

```
public static void testExecuteReportRaw(EVQueryServiceStub stub) {
    try {
        ExecuteReportRawDocument reqEnvelope =
            ExecuteReportRawDocument.Factory.newInstance();
        ExecuteReportRawRequest request =
            reqEnvelope.addNewExecuteReportRaw().addNewRequest();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setPageLength(100);
        request.setRecordStart(1);
        request.setRecordCount(120);
        request.setPersistHandle(ServiceClientHelper.randStr);
        // this query exists on all sites.
        request.setReportId(ServiceClientHelper.STANDARD_REPORT);
        ExecuteReportRawResponseDocument resEnvelope =
stub.executeReportRaw(reqEnvelope);
        ExecuteReportRawResponse response =
            resEnvelope.getExecuteReportRawResponse().getReturn();
        if (response.getSuccess()) {
```

```
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                               response.getReturnMessage());
            System.out.println("XML Report:n" + response.getXmlReport());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                               response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

executeUserSearch

This action retrieves a list of users stored within ExtraView.

Input

Class	Name	Type	Required	Details
ExecuteUserSearchRequest	userId	String	Yes	The callers user name
ExecuteUserSearchRequest	password	String	Yes	The callers password
ExecuteUserSearchRequest	disabled	String	No	Include disabled users in results? { **Y** (yes), **N** (no), **ONLY** (only disabled users)}
ExecuteUserSearchRequest	filter	String	No	This allows your to perform a wildcard pattern search for specific user records. The wildcard is * and you may have more then one of then in the pattern. (eg. ***OB*** or ***OB***)
ExecuteUserSearchRequest	filter_type	String	No	If a filter is given, a filter_type must be given. Possible values ID, FIRST or LAST. This is the field you want the filter to work on.

Output

Class	Name	Type	Required	Details
ExecuteUserSearchResponse	success	boolean	Yes	True is succeeded False if failed
ExecuteUserSearchResponse	returnCode	String	No	See Appendix for details
ExecuteUserSearchResponse	returnMessage	String	No	Human readable message
ExecuteUserSearchResponse	users	Array[]	No	A list of UserBeans
User	userId	String	Yes	
User	userPassword	String	No	
User	firstName	String	No	
User	lastName	String	No	
User	password	String	No	
User	email	String	No	
User	jobTitle	String	No	
User	companyName	String	No	
User	addressLine1	String	No	

Example

```
public static void testExecuteUserSearch(EVReportServiceStub stub) {
    try {
        ExecuteUserSearchDocument reqEnvelope =
            ExecuteUserSearchDocument.Factory.newInstance();
        ExecuteUserSearchRequest request =
            reqEnvelope.addNewExecuteUserSearch().addNewRequest();
        request.setFilter("*" + ServiceClientHelper.ADMIN_USER_ID.toUpperCase() + "*");
        request.setFilterType("ID");
        request.setDisabled("Y");
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        ExecuteUserSearchResponseDocument resEnvelope =
stub.executeUserSearch(reqEnvelope);
        ExecuteUserSearchResponse response =
            resEnvelope.getExecuteUserSearchResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getAllUserRoles

This action gets all the possible roles available to all users within the ExtraView database.

Input

Class	Name	Type	Required	Details
GetAllUserRolesRequest	userId	String	Yes	The callers user name
GetAllUserRolesRequest	password	String	Yes	The callers password

Output

Class	Name	Type	Required	Details

GetAllUserRolesResponse	success	boolean	Yes	True is succeeded False if failed
GetAllUserRolesResponse	returnCode	String	No	See Appendix for details
GetAllUserRolesResponse	returnMessage	String	No	Human readable message
GetAllUserRolesResponse	*roles*	Array[]	No	A list of RoleBeans
Role	roleId	String	Yes	
Role	currentRole	boolean	Yes	
Role	roleName	String	Yes	

Example

```
public static void testGetUserRolesByUser(EVUserServiceStub stub) {
    try {
        GetAllUserRolesDocument reqEnvelope =
GetAllUserRolesDocument.Factory.newInstance();
        GetAllUserRolesRequest request =
reqEnvelope.addNewGetAllUserRoles().addNewRequest();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetAllUserRolesResponseDocument resEnvelope = stub.getAllUserRoles(reqEnvelope);
        GetAllUserRolesResponse response =
            resEnvelope.getGetAllUserRolesResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getAllowedFieldValues

This action retrieves the list of child values for a given parent value when two fields are in an allowed value relationship within the database.

Input

Class	Name	Type	Required	Details
GetAllowedFieldsRequest	userId	String	Yes	The callers user name
GetAllowedFieldsRequest	password	String	Yes	The callers password
GetAllowedFieldsRequest	fieldId	String	Yes	
GetAllowedFieldsRequest	parentFieldId	String	Yes	
GetAllowedFieldsRequest	parentValue	String	Yes	

Output

Class	Name	Type	Required	Details
GetAllowedFieldsResponse	success	boolean	Yes	True is succeeded False if failed
GetAllowedFieldsResponse	returnCode	String	No	See Appendix for details
GetAllowedFieldsResponse	returnMessage	String	No	Human readable message
GetAllowedFieldsResponse	*fields*	Array	No	The list of Field objects
Field	id	String	Yes	

Field	name	String	Yes
Field	value	String	No
Field	childOfFieldId	String	No
Field	repeatingRowField	boolean	No
Field	textAreaField	boolean	No
Field	typeOfUserField	boolean	No

Example

```
public static void testGetAllowedFieldValues(EVSystemServiceStub stub) {
    try {
        GetAllowedFieldValuesDocument reqEnvelope =
GetAllowedFieldValuesDocument.Factory.newInstance();
        GetAllowedFieldsRequest request =
            reqEnvelope.addNewGetAllowedFieldValues().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setFieldId("PROJECT");
        request.setParentFieldId("AREA");
        request.setParentValue("1");
        GetAllowedFieldValuesResponseDocument resEnvelope =
            stub.getAllowedFieldValues(reqEnvelope);
        GetAllowedFieldsResponse response =
            resEnvelope.getGetAllowedFieldValuesResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getBehaviorSetting

This action retrieves the value of a single behavior setting from within the ExtraView database.

Input

Class	Name	Type	Required	Details
GetAppDefaultRequest	userId	String	Yes	The callers user name
GetAppDefaultRequest	password	String	Yes	The callers password
GetAppDefaultRequest	behaviorSettingName	String	Yes	The field name you want the default values of

Output

Class	Name	Type	Required	Details
GetAppDefaultResponse	success	boolean	Yes	True is succeeded False if failed
GetAppDefaultResponse	returnCode	String	No	See Appendix for details
GetAppDefaultResponse	returnMessage	String	No	Human readable message
GetAppDefaultResponse	*field*	Object	No	A single Field object
Field	id	String	Yes	
Field	name	String	No	
Field	value	String	No	

Field	childOfFieldId	String	No
Field	repeatingRowField	boolean	No
Field	textAreaField	boolean	No
Field	typeOfUserField	boolean	No

Example

```
public static void testGetAppDefault(EVSystemServiceStub stub) {
    try {
        GetAppDefaultDocument reqEnvelope = GetAppDefaultDocument.Factory.newInstance();
        GetAppDefaultRequest request = reqEnvelope.addNewGetAppDefault().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setAppDefaultName("DEFAULT_TEXT_REPORT_DELIMITER");
        GetAppDefaultResponseDocument resEnvelope = stub.getAppDefault(reqEnvelope);
        GetAppDefaultResponse response =
resEnvelope.getGetAppDefaultResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getChangedItems

The getChangedItems command returns all the changes to item records, from a specified point in time to the current time.

Input

Class	Name	Type	Required	Details
ItemHistoryRequest	userId	String	Yes	The callers user name

Output

Class	Name	Type	Required	Details
ItemHistoryResponse	success	boolean	Yes	True is succeeded False if failed
ItemHistoryResponse	returnCode	String	No	See Appendix for details
ItemHistoryResponse	returnMessage	String	No	Human readable message
ItemHistoryResponse	*xml*	String	No	A list of AttachmentBeans
ItemHistoryResponse	itemRecords	Array []	No	The date the attachment was added to the item
ItemRecord	numberOfItemRecordFields	int	Yes	The number of field records in the child array
ItemRecord	numberOfItemRecordFields	int	Yes	An array of ItemRecordField objects
ItemRecordField	fieldId	String	Yes	The id of the field
ItemRecordField	fieldTitle	String	Yes	The title of the field
ItemRecordField	fieldValue	String	Yes	The value of the field
ItemRecordField	row	int	Yes	The repeating row value

Example

```
public static void testGetChangedItems(EVItemServiceStub stub) {
    try {
        GetChangedItemsDocument reqEnvelope =
GetChangedItemsDocument.Factory.newInstance();
        ItemHistoryRequest request =
reqEnvelope.addNewGetChangedItems().addNewRequest();
        Calendar startCal = new GregorianCalendar();
        startCal.add(Calendar.DATE, -30);
        // look back 30 days ago
        Calendar endCal = new GregorianCalendar();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setStartTimeSlice(startCal);
        request.setEndTimeSlice(endCal);
        request.setReturnRawXML(true);
        GetChangedItemsResponseDocument resEnvelope = stub.getChangedItems(reqEnvelope);
        ItemHistoryResponse response =
resEnvelope.getGetChangedItemsResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                response.getReturnMessage());
            System.out.println("XML Report:n" + response.getXml());
            for (ItemRecord record: response.getItemRecordsArray()) {
                System.out.println(" ++++++++++++++ ");
                System.out.println(" + New Record + ");
                for (ItemRecordField field: record.getItemRecordFieldsArray()) {
                    System.out.println(" + New Field + ");
                    System.out.println(" - Field Id: " + field.getFieldId());
                    System.out.println(" - Value: " + field.getFieldValue());
                    System.out.println(" - Title: " + field.getFieldTitle());
                    System.out.println(" - Row: " + field.getRow());
                }
            }
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                              response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getFieldList

This action provides a list of all the available fields to the user who calls the function.

Input

Class	Name	Type	Required	Details
GetFieldListRequest	userId	String	Yes	The callers user name

GetFieldListRequest	password	String	Yes	The callers password

Output

Class	Name	Type	Required	Details
GetFieldListResponse	success	boolean	Yes	True is succeeded False if failed
GetFieldListResponse	returnCode	String	No	See Appendix for details
GetFieldListResponse	returnMessage	String	No	Human readable message
GetFieldListResponse	*fields*	Array	No	A list of Field objects
Field	id	String	Yes	
Field	name	String	Yes	
Field	value	String	No	
Field	childOfFieldId	String	No	
Field	repeatingRowField	boolean	No	
Field	textAreaField	boolean	No	
Field	typeOfUserField	boolean	No	

Example

```
public static void testGetFieldList(EVReportServiceStub stub) {
    try {
        GetFieldListDocument reqEnvelope = GetFieldListDocument.Factory.newInstance();
        GetFieldListRequest request = reqEnvelope.addNewGetFieldList().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetFieldListResponseDocument resEnvelope = stub.getFieldList(reqEnvelope);
        GetFieldListResponse response =
resEnvelope.getGetFieldListResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getChangedItems

The getChangedItems command returns all the changes to item records, from a specified point in time to the current time.

Input

Class	Name	Type	Required	Details
ItemHistoryRequest	userId	String	Yes	The caller's user name
ItemHistoryRequest	password	String	Yes	The caller's password
ItemHistoryRequest	returnRawXML	Boolean	No	If you want the raw XML returned with the parse fields. This should be set to false for performance.
ItemHistoryRequest	startTimeSlice	Date	Yes	The cutoff timestamp value is not included in the issues generated by this command. The comparison is “item timestamp is greater than cutoff timestamp”.

ItemHistoryRequest	endTimeSlice	Date	Yes	The optional userNameMask may be used to override the behavior setting named USERNAME_DISPLAY, for the duration of the execution of a single API call. This allows the developer to return the user names in a different format than the system-wide default.
ItemHistoryRequest	userNameMask	String	No	The optional userNameMask may be used to override the behavior setting named USERNAME_DISPLAY, for the duration of the execution of a single API call. This allows the developer to return the user names in a different format than the system-wide default.
ItemHistoryRequest	filters	Array[]	No	List of filters for the history.
Field	id	String	Yes	
Field	name	String	No	
Field	value	String	No	
Field	childOfFieldId	String	No	
Field	repeatingRowField	boolean	No	
Field	textAreaField	boolean	No	
Field	typeOfUserField	boolean	No	

Output

Class	Name	Type	Required	Details
ItemHistoryResponse	success	boolean	Yes	True is succeeded False if failed
ItemHistoryResponse	returnCode	String	No	See Appendix for details
ItemHistoryResponse	returnMessage	String	No	Human readable message
ItemHistoryResponse	xml	String	No	A list of AttachmentBeans
ItemHistoryResponse	itemRecords	Array[]	No	The date the attachment was added to the item
ItemRecord	numberOfItemRecordFields	int	Yes	The number of field records in the child array
ItemRecord	itemRecordFields	Array[]	Yes	An array of ItemRecordField objects
ItemRecordField	fieldId	String	Yes	The id of the field
ItemRecordField	fieldTitle	String	Yes	The title of the field
ItemRecordField	fieldValue	String	Yes	The value of the field
ItemRecordField	row	int	Yes	The repeating row value

Example

```
public static void testGetChangedItems(EVItemServiceStub stub) {
    try {
        GetChangedItemsDocument reqEnvelope =
GetChangedItemsDocument.Factory.newInstance();
        ItemHistoryRequest request =
reqEnvelope.addNewGetChangedItems().addNewRequest();
        Calendar startCal = new GregorianCalendar();
        startCal.add(Calendar.DATE, -30);
        // look back 30 days ago
        Calendar endCal = new GregorianCalendar();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setStartTimeSlice(startCal);
        request.setEndTimeSlice(endCal);
        request.setReturnRawXML(true);
        GetChangedItemsResponseDocument resEnvelope = stub.getChangedItems(reqEnvelope);
        ItemHistoryResponse response =
resEnvelope.getGetChangedItemsResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                               response.getReturnMessage());
            System.out.println("XML Report:n" + response.getXml());
            for (ItemRecord record: response.getItemRecordsArray()) {
                System.out.println(" ++++++++++++++ ");
                System.out.println(" + New Record + ");
                for (ItemRecordField field: record.getItemRecordFieldsArray()) {
                    System.out.println(" + New Field + ");
                    System.out.println(" - Field Id: " + field.getFieldId());
                    System.out.println(" - Value: " + field.getFieldValue());
                    System.out.println(" - Title: " + field.getFieldTitle());
                    System.out.println(" - Row: " + field.getRow());
                }
            }
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                               response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getFieldMetaData

This function retrieved the metadata related to the fields and their fixed values.

Input

Class	Name	Type	Required	Details
GetFieldMetaDataRequest	userId	String	Yes	The callers user name

GetFieldMetaDataRequest	Password	String	Yes	The callers password
GetFieldMetaDataRequest	fieldIds	String[]	No	The array of field id's the meta data is requested for.
GetFieldMetaDataRequest	getAllFields	boolean	No	An overriding flag to return all the possible fields.

Output

Class	Name	Type	Required	Details
GetFieldMetaDataResponse	Success	boolean	Yes	True is succeeded False if failed
GetFieldMetaDataResponse	returnCode	String	No	See Appendix for details
GetFieldMetaDataResponse	returnMessage	String	No	Human readable message
GetFieldMetaDataResponse	FieldMetaData	String	Yes	The report in XML format

Example

```
public static void getFieldMetaData(EVSystemServiceStub stub) {
    try {
        GetFieldMetaDataDocument reqEnvelope =
            GetFieldMetaDataDocument.Factory.newInstance();
        GetFieldMetaDataRequest request =
            reqEnvelope.addNewGetFieldMetaData().addNewRequest();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setGetAllFields(true);
        GetFieldMetaDataResponseDocument resEnvelope =
stub.getFieldMetaData(reqEnvelope);
        GetFieldMetaDataResponse response =
            resEnvelope.getGetFieldMetaDataResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
```

```
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

getFieldTitle

This action retrieves the title of a field in the ExtraView data dictionary, by providing its fixed name. Other details about the field are also returned, as shown below in the output of the call.

Input

Class	Name	Type	Required	Details
GetFieldTitleRequest	userId	String	Yes	The callers user name
GetFieldTitleRequest	password	String	Yes	The callers password
GetFieldTitleRequest	fieldId	String	Yes	The field id you want to get the title of

Output

Class	Name	Type	Required	Details
GetFieldTitleResponse	success	boolean	Yes	True is succeeded False if failed
GetFieldTitleResponse	returnCode	String	No	See Appendix for details
GetFieldTitleResponse	returnMessage	String	No	Human readable message
GetFieldTitleResponse	*field*	Object	No	A single Field object

Field	id	String	Yes
Field	name	String	Yes
Field	value	String	No
Field	childOfFieldId	String	No
Field	repeatingRowField	boolean	No
Field	textAreaField	boolean	No
Field	typeOfUserField	boolean	No

Example

```
public static void testGeFieldTitle(EVSystemServiceStub stub) {
    try {
        GetFieldTitleDocument reqEnvelope = GetFieldTitleDocument.Factory.newInstance();
        GetFieldTitleRequest request = reqEnvelope.addNewGetFieldTitle().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setFieldId("STATUS");
        GetFieldTitleResponseDocument resEnvelope = stub.getFieldTitle(reqEnvelope);
        GetFieldTitleResponse response =
resEnvelope.getGetFieldTitleResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getFields

This action provides a list of all the fields available to the user. Note that all of ExtraView's security is in

force and the calling user will only see the fields to which he has access. Also note that there is no difference in the way that User Defined Fields (UDF's) are shown compared to ExtraView's built-in fields. These UDF's are handled in a seamless way within the API.

Input

Class	Name	Type	Required	Details
GetFieldsRequest	userId	String	Yes	The callers user name
GetFieldsRequest	password	String	Yes	The callers password

Output

Class	Name	Type	Required	Details
GetFieldsResponse	success	boolean	Yes	True is succeeded False if failed
GetFieldsResponse	returnCode	String	No	See Appendix for details
GetFieldsResponse	returnMessage	String	No	Human readable message
GetFieldsResponse	fields	Array[]	No	A list of Field objects
Field	id	String	Yes	
Field	name	String	No	
Field	value	String	No	
Field	childOfFieldId	String	No	

Field	repeatingRowField	boolean	No
Field	textAreaField	boolean	No
Field	typeOfUserField	boolean	No

Example

```
public static void testGetFields(EVSystemServiceStub stub) {
    try {
        GetFieldsDocument reqEnvelope = GetFieldsDocument.Factory.newInstance();
        GetFieldsRequest request = reqEnvelope.addNewGetFields().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetFieldsResponseDocument resEnvelope = stub.getFields(reqEnvelope);
        GetFieldsResponse response = resEnvelope.getGetFieldsResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getHeartbeat

This API call provides an indication of the status of ExtraView. It is often used on a server remote to the ExtraView installation and called every few minutes to check that the ExtraView server is functioning correctly. If the call does not get a valid return, an alarm or notification may then be sent to an administrator.

Input

Class	Name	Type	Required	Details
GetHeartbeatRequest	userId	String	Yes	The callers user name

GetHeartbeatRequest	password	String	Yes	The callers password

Output

Class	Name	Type	Required	Details
GetHeartbeatResponse	success	boolean	Yes	True is succeeded False if failed
GetHeartbeatResponse	returnCode	String	No	See Appendix for details
GetHeartbeatResponse	returnMessage	String	No	Human readable message
GetHeartbeatResponse	evStatus	String	Yes	ExtraView System Status
GetHeartbeatResponse	dbStatus	String	Yes	ExtraView Database Status
GetHeartbeatResponse	dbDate	Date	Yes	The current date in the Database
GetHeartbeatResponse	freeMemory	String	Yes	Free memory on application server
GetHeartbeatResponse	totalMemory	Integer	Yes	Total memory on application server
GetHeartbeatResponse	serviceCount	Integer	Yes	The current number of Extraview sessions
GetHeartbeatResponse	executeTime	Long	Yes	The time it took to execute this call

GetHeartbeatResponse	tasks	Array of TaskInfo	No	The information on tasks configured on the node of the server where the service is executed
TaskInfo	nodeId	String	No	The name of the node hosting the service
TaskInfo	startOption	String	No	
TaskInfo	taskState	String	No	START_NOW, STOP_NOW, START_ON_BOOT, or none
TaskInfo	pollInterval	String	No	The minimum number of seconds between polled executions
TaskInfo	threads	Array of ThreadInfo	No	
ThreadInfo	threadState	String	No	Running or stopped
ThreadInfo	secsSinceExecution	Long	No	The number of seconds since the task was most recently scheduled to run
ThreadInfo	priority	Long	No	The priority of the thread (using Java thread priority values). This may not appear in the output

Example

```
public static void testGetHeartbeat(EVSystemServiceStub stub) {
    try {
        GetHeartbeatDocument reqEnvelope = GetHeartbeatDocument.Factory.newInstance();
```

```
        GetHeartbeatRequest request = reqEnvelope.addNewGetHeartbeat().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetHeartbeatResponseDocument resEnvelope = stub.getHeartbeat(reqEnvelope);
        GetHeartbeatResponse response =
resEnvelope.getGetHeartbeatResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getItem

Retrieves an item from the ExtraView database based upon the itemId you provide in the function call.

Input

Class	Name	Type	Required	Details
GetItemRequest	userId	String	Yes	The callers user name
GetItemRequest	password	String	Yes	The callers password
GetItemRequest	itemId	Integer	No	The Id of the item being fetched
GetItemRequest	itemId	int	?	The int item id (this is optional if itemIdStr is given, else it is required)
GetItemRequest	returnRawXML	Boolean	No	If you want the raw XML returned with the parse fields. This should be set to false for performance.

Output

Class	Name	Type	Required	Details
GetItemResponse	success	boolean	Yes	True is succeeded False if failed
GetItemResponse	returnCode	String	No	See Appendix for details
GetItemResponse	returnMessage	String	No	Human readable message
GetItemResponse	xml	String	No	The item in XML format
ExecuteItemSearchResponse	itemRecords	Array[]	Yes	An array of ItemRecord objects
ItemRecord	numberOfItemRecordFields	int	Yes	The number of field records in the child array
ItemRecord	itemRecordFields	Array[]	Yes	An array of ItemRecordField objects
ItemRecordField	fieldId	String	Yes	The id of the field
ItemRecordField	fieldTitle	String	Yes	The title of the field
ItemRecordField	fieldValue	String	Yes	The value of the field
ItemRecordField	row	int	Yes	The repeating row value

Example

```
public static void testGetItem(EVItemServiceStub stub) {
    try {
        GetItemDocument reqEnvelope = GetItemDocument.Factory.newInstance();
        GetItemRequest request = reqEnvelope.addNewGetItem().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setItemId(CREATED_ISSUE_ID);
        request.setReturnRawXML(true);
        GetItemResponseDocument resEnvelope = stub.getItem(reqEnvelope);
        GetItemResponse response = resEnvelope.getGetItemResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getItemAttachments

This function returns the metadata about the file attachments for an item that exists in ExtraView. This method will only return data about the attachments. To get the attachments use the http call to the API.

Input

Class	Name	Type	Required	Details
GetItemAttachmentsRequest	userId	String	Yes	The callers user name
GetItemAttachmentsRequest	password	String	Yes	The callers password
GetItemAttachmentRequest	itemId	int	?	The int item id. This field is optional if itemIdStr is given, else it is required. It is recommended that you use itemIdStr
GetItemAttachmentRequest	itemIdStr	String	?	The String item Id. This field is optional if itemId is used. It is

				recommended that you use itemIdStr

Output

Class	Name	Type	Required	Details
GetItemAttachmentsResponse	success	boolean	Yes	True is succeeded False if failed
GetItemAttachmentsResponse	returnCode	String	No	See Appendix for details
GetItemAttachmentsResponse	returnMessage	String	No	Human readable message
GetItemAttachmentsResponse	*attachments*	Array	No	A list of AttachmentBeans
Attachment	attachmentDate	Date	Yes	The date the attachment was added to the item
Attachment	fileName	String	Yes	The file name of the attachment
Attachment	fileSize	Integer	Yes	The file size of the attachment
Attachment	attachmentCreater	String	Yes	The userId who insertd the attachment
Attachment	attachmentId	Integer	Yes	The id of the attachment
Attachment	description	String	Yes	The description of the attachment

Attachment	fileType	String	No	The optional mime type

Example

```
public static void testGetItemAttachments(EVItemServiceStub stub) {
    try {
        GetItemAttachmentsDocument reqEnvelope =
GetItemAttachmentsDocument.Factory.newInstance();
        GetItemAttachmentsRequest request =
            reqEnvelope.addNewGetItemAttachments().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setItemId(CREATED_ISSUE_ID);
        GetItemAttachmentsResponseDocument resEnvelope =
            stub.getItemAttachments(reqEnvelope);
        GetItemAttachmentsResponse response =
            resEnvelope.getGetItemAttachmentsResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getItemAudits

The getItemAudits actionreturns all the changes to a filtered list of item records, from a specified point in time to the current time.

Input

Class	Name	Type	Required	Details
ItemHistoryRequest	userId	String	Yes	The caller's user ID
ItemHistoryRequest	password	String	Yes	The caller's password
ItemHistoryRequest	returnRawXML			

ItemHistoryRequest	startTimeSlice	Date	Yes	The cutoff timestamp value is not included in the issues generated by this command; that is, the comparison is *item timestamp is greater than cutoff timestamp*
ItemHistoryRequest	endTimeSlice	Date	No	The endTimeSlice is optional. If it is omitted, there is no constraint on the end of the cutoff period. If it is provided, the value must be greater than the value of cutoff. This is used to limit the items for which history is generated
ItemHistoryRequest	userNameMask	String	No	The optional userNameMask may be used to override the behavior setting named USERNAME_DISPLAY, for the duration of the execution of a single API call. This allows the developer to return the user names in a different format than the system-wide default stored in the behavior setting
ItemHistoryRequest	filters	Array[]	No	List of filters for the history
Field	id	boolean	No	
Field	name	boolean	No	
Field	value	boolean	No	
Field	childOfFieldId	boolean	No	
Field	repeatingRowField	boolean	No	

Field	textAreaField	boolean	No
Field	typeOfUserField	boolean	No

Output

Class	Name	Type	Required	Details
ItemHistoryResponse	success	boolean	Yes	True is succeeded False if failed
ItemHistoryResponse	returnCode	String	No	See Appendix for details
ItemHistoryResponse	returnMessage	String	No	Human readable message
ItemHistoryResponse	*xml*	String	No	The number of field records in the child array
ItemHistoryResponse	itemRecords	Array[]	Yes	An array of itemRecords objects
ItemRecord	numberOfItemRecordFields	int	Yes	The number of field records in the child array
ItemRecord	itemRecordFields	Array[]	Yes	An array of ItemRecordField objects
ItemRecordField	fieldId	String	Yes	The id of the field
ItemRecordField	fieldTitle	String	Yes	The title of the field

ItemRecordField	fieldValue	String	Yes	The value of the field
ItemRecordField	row	boolean	Yes	The repeating row value

Example

```
public static void testGetItemAudits(EVItemServiceStub stub) {
    try {
        GetItemAuditsDocument reqEnvelope = GetItemAuditsDocument.Factory.newInstance();
        ItemHistoryRequest request = reqEnvelope.addNewGetItemAudits().addNewRequest();
        Calendar startCal = new GregorianCalendar();
        startCal.add(Calendar.DATE, -30);
        // look back 30 days ago
        Calendar endCal = new GregorianCalendar();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setStartTimeSlice(startCal);
        request.setEndTimeSlice(endCal);
        request.setReturnRawXML(true);
        GetItemAuditsResponseDocument resEnvelope = stub.getItemAudits(reqEnvelope);
        ItemHistoryResponse response =
resEnvelope.getGetItemAuditsResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                              response.getReturnMessage());
            System.out.println("XML Report:n" + response.getXml());
            for (ItemRecord record: response.getItemRecordsArray()) {
                System.out.println(" ++++++++++++++ ");
                System.out.println(" + New Record + ");
                for (ItemRecordField field: record.getItemRecordFieldsArray()) {
                    System.out.println(" + New Field + ");
                    System.out.println(" - Field Id: " + field.getFieldId());
                    System.out.println(" - Value: " + field.getFieldValue());
                    System.out.println(" - Title: " + field.getFieldTitle());
                    System.out.println(" - Row: " + field.getRow());
                }
            }
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                              response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getItemFieldList

This action provides a list of all the available fields to the user. Note that all of ExtraView's security is in force and an individual user will only see the fields to which he has access. Also note that there is no difference in the way that User Defined Fields (UDF's) are shown than other fields. UDF's are handled in a

seamless way within the API.

Input

Class	Name	Type	Required	Details
GetItemFieldListRequest	userId	String	Yes	The callers user name
GetItemFieldListRequest	password	String	Yes	The callers password

Output

Class	Name	Type	Required	Details
GetItemFieldListResponse	success	boolean	Yes	True is succeeded False if failed
GetItemFieldListResponse	returnCode	String	No	See Appendix for details
GetItemFieldListResponse	returnMessage	String	No	Human readable message
GetItemFieldListResponse	*fields*	Array	No	A list of FieldBeans
Field	id	String	Yes	
Field	name	String	Yes	
Field	value	String	No	
Field	childOfFieldId	String	No	

Field	repeatingRowField	boolean	No
Field	textAreaField	boolean	No
Field	typeOfUserField	boolean	No

Example

```
public static void testGetItemFieldList(EVItemServiceStub stub) {
    try {
        GetItemFieldListDocument reqEnvelope =
GetItemFieldListDocument.Factory.newInstance();
        GetItemFieldListRequest request =
            reqEnvelope.addNewGetItemFieldList().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetItemFieldListResponseDocument resEnvelope =
stub.getItemFieldList(reqEnvelope);
        GetItemFieldListServiceResponse response =
            resEnvelope.getGetItemFieldListResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getPing

This action returns a web services status independent of the ExtraView application.

Input

There are no inputs for this action.

Output

There are no outputs for this action. Independent of any user ID, password or any other parameter, this action confirms that a connection to the server exists and is functioning.

Example

http://SERVER_NAME/EV_NAME/services/EVSystemService/getPing?

or

```
public static void systemStatus(EVSystemServiceStub stub) {
    try {
        GetPingResponseDocument response = stub.getPing();
        System.out.println(response.getGetPingResponse().getReturn());
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println("nnn");
    }
}
```

getReportHandle

- Add new comment

The getReportHandle command returns all the report handles available to the calling user.

Input

Class	Name	Type	Required	Details
GetReportHandleRequest	userId	String	Yes	The caller's user name
GetReportHandleRequest	password	String	Yes	The caller's password

Output

Class	Name	Type	Required	Details
GetReportHandleResponse	success	boolean	Yes	True is succeeded False if failed
GetReportHandleResponse	returnCode	String	No	See Appendix for details
GetReportHandleResponse	returnMessage	String	No	Human readable message

GetReportHandleResponse	reportHandles	Array[]	No	The date the attachment was added to the item
ReportHandle	reportId	int	Yes	The report Id
ReportHandle	owner	String	Yes	PUBLIC, PRIVATE or UNKNOWN
ReportHandle	reportType	String	Yes	See the Appendix on Report Type Table
ReportHandle	ReportTitle	String	Yes	The title of the report
ReportHandle	reportDesc	String	Yes	The description of the report

Example

```
public static void testGetReportHandle(EVQueryServiceStub stub) {
    try {
        GetReportHandleDocument reqEnvelope =
GetReportHandleDocument.Factory.newInstance();
        GetReportHandleRequest request =
reqEnvelope.addNewGetReportHandle().addNewRequest();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetReportHandleResponseDocument resEnvelope = stub.getReportHandle(reqEnvelope);
        GetReportHandleResponse response =
            resEnvelope.getGetReportHandleResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
            for (ReportHandle reportHandle: response.getReportHandlesArray()) {
                System.out.println(" +++++++++++++++++ ");
                System.out.println(" Report Handle: ");
                System.out.println(" - Report Id: " + reportHandle.getReportId());
                System.out.println(" - Owner: " + reportHandle.getOwner());
                System.out.println(" - Type: " + reportHandle.getReportType());
                System.out.println(" - Title: " + reportHandle.getReportTitle());
                System.out.println(" - Desc: " + reportHandle.getReportDesc());
            }
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
```

```
        System.err.println("nnn");
    }
}
```

getUserByEmail

This action retrieves the first found user within ExtraView for a given email address. Note that if the same email address is used for different users, only the first is returned.

Input

Class	Name	Type	Required	Details
GetUserByEmailRequest	userId	String	Yes	The callers user name
GetUserByEmailRequest	password	String	Yes	The callers password
GetUserByEmailRequest	email	String	Yes	The email address of the user your getting

Output

Class	Name	Type	Required	Details
GetUserByEmailResponse	success	boolean	Yes	True is succeeded False if failed
GetUserByEmailResponse	returnCode	String	No	See Appendix for details
GetUserByEmailResponse	returnMessage	String	No	Human readable message
GetUserByEmailResponse	user	Object	No	A Single UserBean Object

UserBean	userId	String	Yes
UserBean	userPassword	String	No
UserBean	firstName	String	No
UserBean	lastName	String	No
UserBean	password	String	No
UserBean	email	String	No
UserBean	jobTitle	String	No
UserBean	companyName	String	No
UserBean	addressLine1	String	No
UserBean	addressLine2	String	No
UserBean	city	String	No
UserBean	state	String	No
UserBean	postalCode	String	No
UserBean	country	String	No
UserBean	workPhoneNumber	String	No

Example

```
public static void testGetUserByEmail(EVUserServiceStub stub) {
    try {
        GetUserByEmailDocument reqEnvelope =
GetUserByEmailDocument.Factory.newInstance();
        GetUserByEmailRequest request =
reqEnvelope.addNewGetUserByEmail().addNewRequest();
        request.setEmail(ServiceClientHelper.generatedEmail);
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetUserByEmailResponseDocument resEnvelope = stub.getUserByEmail(reqEnvelope);
        GetUserByEmailResponse response =
            resEnvelope.getGetUserByEmailResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                               response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                               response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getUserById

This action retrieves a user's details from ExtraView for the given user ID.

Input

Class	Name	Type	Required	Details
GetUserByIdRequest	userId	String	Yes	The callers user name
GetUserByIdRequest	password	String	Yes	The callers password
GetUserByIdRequest	getUserId	String	Yes	The user Id of the user your getting

Output

Class	Name	Type	Required	Details

GetUserByIdResponse	success	boolean	Yes	True is succeeded False if failed
GetUserByIdResponse	returnCode	String	No	See Appendix for details
GetUserByIdResponse	returnMessage	String	No	Human readable message
GetUserByIdResponse	user	Object	No	A Single UserBean Object
UserBean	userId	String	Yes	
UserBean	userPassword	String	No	
UserBean	firstName	String	No	
UserBean	lastName	String	No	
UserBean	password	String	No	
UserBean	email	String	No	
UserBean	jobTitle	String	No	
UserBean	companyName	String	No	
UserBean	addressLine1	String	No	
UserBean	addressLine2	String	No	

UserBean	city	String	No
UserBean	state	String	No
UserBean	postalCode	String	No
UserBean	country	String	No
UserBean	workPhoneNumber	String	No
UserBean	homePhoneNumber	String	No
UserBean	cellPhoneNumber	String	No
UserBean	faxNumber	String	No
UserBean	pagerNumber	String	No

Example

```
public static void testGetUserById(EVUserServiceStub stub) {
    try {
        GetUserByIdDocument reqEnvelope = GetUserByIdDocument.Factory.newInstance();
        GetUserByIdRequest request = reqEnvelope.addNewGetUserById().addNewRequest();
        request.setGetUserId(ServiceClientHelper.generatedUserId);
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetUserByIdResponseDocument resEnvelope = stub.getUserById(reqEnvelope);
        GetUserByIdResponse response = resEnvelope.getGetUserByIdResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
```

```
}
```

getUserFields

This action retrieves a set of user fields from within ExtraView.

Input

Class	Name	Type	Required	Details
GetUserFieldsRequest	userId	String	Yes	The callers user name
GetUserFieldsRequest	password	String	Yes	The callers password

Output

Class	Name	Type	Required	Details
GetUserFieldsResponse	success	boolean	Yes	True is succeeded False if failed
GetUserFieldsResponse	returnCode	String	No	See Appendix for details
GetUserFieldsResponse	returnMessage	String	No	Human readable message
GetUserFieldsResponse	*userFields*	Array[]	No	A list of Field objects
Field	id	String	Yes	
Field	name	String	No	
Field	value	String	No	

Example

```
public static void testGetUserFields(EVUserServiceStub stub) {
    try {
        GetUserFieldsDocument reqEnvelope = GetUserFieldsDocument.Factory.newInstance();
        GetUserFieldsRequest request = reqEnvelope.addNewGetUserFields().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetUserFieldsResponseDocument resEnvelope = stub.getUserFields(reqEnvelope);
        GetUserFieldsResponse response =
resEnvelope.getGetUserFieldsResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getUserRoles

This action retrieves a list of the roles for an existing user within ExtraView.

Input

Class	Name	Type	Required	Details
GetUserRolesRequest	userId	String	Yes	The callers user name
GetUserRolesRequest	password	String	Yes	The callers password

Output

Class	Name	Type	Required	Details
GetUserRolesResponse	success	boolean	Yes	True is succeeded False if failed

GetUserRolesResponse	returnCode	String	No	See Appendix for details
GetUserRolesResponse	returnMessage	String	No	Human readable message
GetUserRolesResponse	*roles*	Array[]	No	A list of Role Objects
Role	roleId	String	Yes	
Role	currentRole	boolean	Yes	
Role	roleName	String	Yes	

Example

```
public static void testGetUserRoles(EVUserServiceStub stub) {
    try {
        GetUserRolesDocument reqEnvelope = GetUserRolesDocument.Factory.newInstance();
        GetUserRolesRequest request = reqEnvelope.addNewGetUserRoles().addNewRequest();
        request.setUserId(ServiceClientHelper.generatedUserId);
        request.setPassword(ServiceClientHelper.generatedPassword);
        GetUserRolesResponseDocument resEnvelope = stub.getUserRoles(reqEnvelope);
        return resEnvelope.getGetUserRolesResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                 response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

getVersion

This action returns the build information of ExtraView. The build information identifies the specific build and version number of the ExtraView server.

Input

Class	Name	Type	Required	Details
GetVersionRequest	userId	String	Yes	The callers user name
GetVersionRequest	password	String	Yes	The callers password

Output

Class	Name	Type	Required	Details
GetVersionResponse	success	boolean	Yes	True is succeeded False if failed
GetVersionResponse	returnCode	String	No	See Appendix for details
GetVersionResponse	returnMessage	String	No	Human readable message
GetVersionResponse	revision	String	Yes	The revision of ExtraView
GetVersionResponse	modificationData	String	Yes	The last modification date of the ExtraView code base.

Example

```
public static void testGetVersion(EVSystemServiceStub stub) {
    try {
        // first we build the request...
        GetVersionDocument reqEnvelope = GetVersionDocument.Factory.newInstance();
        GetVersionRequest request = reqEnvelope.addNewGetVersion().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        GetVersionResponseDocument resEnvelope = stub.getVersion(reqEnvelope);
        GetVersionResponse response = resEnvelope.getGetVersionResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                               response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                               response.getReturnMessage());
        }
```

```
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

insertItem

This API call inserts a new record into the ExtraView database. All fields are treated as optional, and all defined business rules are executed and checked before and after the record is inserted.

Input

Class	Name	Type	Required	Details
InsertItemRequest	userId	String	Yes	The callers user name
InsertItemRequest	password	String	Yes	The callers password
InsertItemRequest	sendEmail	boolean	Yes	
InsertItemRequest	itemFields	Array[]	Yes	An array of ItemRecordField objects
ItemRecordField	fieldId	String	Yes	
ItemRecordField	fieldTitle	String	Yes	
ItemRecordField	fieldValue	String	No	
ItemRecordField	row	int	No	

Optional parameters for handling document and image files:

Class	Name	Type	Required	Details

ItemFileField	itemImages	Array[]	No	The array of image files and data
ItemFileField	itemDocuments	Array[]	No	The array of document files and data
ItemFileField	ddName	String	No	The data dictionary name of the file
ItemFileField	fileName	String	No	The target file name to be saved
ItemFileField	file	String	No	The file
ItemFileField	description	String	No	The description of the file
ItemFileField	charset	String	No	The charset of the file
ItemFileField	contentType	String	No	The content type - e.g. "image/GIF"
ItemFileField	rowId			The repeating row Id
ItemFileField	rowNumber	int	No	The row number - currently not supported

Output

Class	Name	Type	Required	Details
InsertItemResponse	success	boolean	Yes	True is succeeded False if failed
InsertItemResponse	returnCode	String	No	See Appendix for details

InsertItemResponse	returnMessage	String	No	Human readable message
InsertItemResponse	itemId	String	Yes	The Id of the insertd item

Example

```
public static void testInsertItem(EVItemServiceStub stub) {
    try {
        InsertItemDocument reqEnvelope = InsertItemDocument.Factory.newInstance();
        InsertItemRequest request = reqEnvelope.addNewInsertItem().addNewRequest();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        List items = new ArrayList();
        ItemRecordField item = null;
        for (int i = 0; i < 4; i++) {
            item = ItemRecordField.Factory.newInstance();
            item.setRow(0);
            // no repeating rows for this example
            switch (i) {
                case 0:
                    item.setFieldId("status");
                    item.setFieldValue("NEW");
                    break;
                case 1:
                    item.setFieldId("assigned_to");
                    item.setFieldValue(ServiceClientHelper.generatedUserId);
                    break;
                case 2:
                    item.setFieldId("short_description");
                    item.setFieldValue("This is the short desc or title");
                    break;
                case 3:
                    item.setFieldId("description");
                    item.setFieldValue("this is the long desc");
                    break;
                default:
                    item.setFieldId("unknown");
                    item.setFieldValue("");
                    break;
            }
            items.add(item);
        }
        request.setItemFieldsArray(items.toArray(new ItemRecordField[items.size()]));
        request.setSendEmail(false);
        InsertItemResponseDocument resEnvelope = stub.insertItem(reqEnvelope);
        InsertItemResponse response = resEnvelope.getInsertItemResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                              response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                              response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
```

```
    }
}
```

Example 2 - Handling Document & Image Fields

```
public static void testInsertItem(EVItemServiceStub stub) {
    try {
        File imageFile = new File("./test.GIF");
        DataInputStream dis = new DataInputStream(new FileInputStream(imageFile));
        BufferedReader br = new BufferedReader(new InputStreamReader(dis));
        List bFileList = new ArrayList();
        try {
            byte b;
            while (true) {
                b = dis.readByte();
                bFileList.add(b);
            }
        } catch (Exception e) { /* handle the EOF the lazy way. */
        }
        byte[] fileBytes = new byte[bFileList.size()];
        for (int i = 0; i items = new ArrayList(); ItemRecordField item = null;
        for (int i = 0; i < 4; i++) {
            item = ItemRecordField.Factory.newInstance();
            item.setRow(0);
            // no repeating rows for this example
            switch (i) {
                case 0:
                    item.setFieldId("status");
                    item.setFieldValue("NEW");
                    break;
                case 1:
                    item.setFieldId("assigned_to");
                    item.setFieldValue(ServiceClientHelper.generatedUserId);
                    break;
                case 2:
                    item.setFieldId("SHORT_DESCR");
                    item.setFieldValue("This is the short desc or title");
                    break;
                case 3:
                    item.setFieldId("DESCRIPTION");
                    item.setFieldValue("this is the long desc");
                    break;
                default:
                    item.setFieldId("unknown");
                    item.setFieldValue("");
                    break;
            }
            items.add(item);
        }
        request.setItemFieldsArray(items.toArray(new ItemRecordField[items.size()]));
        request.setSendEmail(false);
        List itemDocumentFields = new ArrayList();
        ItemFileField document = ItemFileField.Factory.newInstance();
        document.setCharset("UTF-8"); document.setContentType("image/gif");
        document.setDescription("This is a test image file in a document field!");
        document.setFile(fileBytes); document.setDdName("DOCUMENT");
        document.setFileName(imageFile.getName()); itemDocumentFields.add(document);
        request.setItemDocumentsArray(itemDocumentFields.toArray(new
            ItemFileField[itemDocumentFields.size()]));
```

```
        InsertItemResponseDocument resEnvelope = stub.insertItem(reqEnvelope);
        InsertItemResponse response = resEnvelope.getInsertItemResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
            System.out.println("Item [" + response.getItemId() + "] was created.");
            CREATED_Item_ID = response.getItemId();
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
        } catch (Exception e) {
            e.printStackTrace();
            System.err.println("\n\n\n");
        }
    }
    public static void testUpdateItem(EVItemServiceStub stub) {
        try {
            File imageFile = new File("./test.GIF");
            DataInputStream dis = new DataInputStream(new FileInputStream(imageFile));
            BufferedReader br = new BufferedReader(new InputStreamReader(dis));
            List bFileList = new ArrayList();
            try {
                byte b;
                while (true) {
                    b = dis.readByte();
                    bFileList.add(b);
                }
            } catch (Exception e) { /* handle the EOF the lazy way. */
            }
            byte[] fileBytes = new byte[bFileList.size()];
            for (int i = 0; i
                items = new ArrayList();
                ItemRecordField item = ItemRecordField.Factory.newInstance();
item.setRow(0);
                // no repeating rows for this example
                item.setFieldId("SHORT_DESCR");
                item.setFieldValue("This is the short desc or title -- changed!");
                items.add(item);
                request.setItemFieldsArray(items.toArray(new
ItemRecordField[items.size()]));
                request.setSendEmail(false);
                List itemImageFields = new ArrayList();
                ItemFileField image = ItemFileField.Factory.newInstance();
                image.setCharset("UTF-8");
                image.setContentType("image/gif");
                image.setDescription("This is a test image file in a image field!");
                image.setFile(fileBytes);
                image.setDdName("IMAGE");
                image.setFileName(imageFile.getName());
                itemImageFields.add(image);
                request.setItemImagesArray(itemImageFields.toArray(new
                    ItemFileField[itemImageFields.size()]));
                UpdateItemResponseDocument resEnvelope = stub.updateItem(reqEnvelope);
                UpdateItemResponse response =
                    resEnvelope.getUpdateItemResponse().getReturn();
            if (response.getSuccess()) {
                System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                    response.getReturnMessage());
                System.out.println("Item [" + response.getItemId() + "] was changed.");
                CREATED_Item_ID = response.getItemId();
            } else {
```

```
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
        } catch (Exception e) {
            e.printStackTrace();
            System.err.println("\n\n\n");
        }
    }
```

insertUser

This action inserts a new user in the ExtraView database. The user must not exist in the database, else an error results.

Input

Class	Name	Type	Required	Details
InsertUserRequest	userId	String	Yes	The callers user name
InsertUserRequest	password	String	Yes	The callers password
InsertUserRequest	newUserId	String	Yes	The desired new user id (must be unique)
InsertUserRequest	newUserPassword	String	Yes	The desired new password
InsertUserRequest	firstName	String	Yes	The new users first name
InsertUserRequest	lastName	String	Yes	The new users last name
InsertUserRequest	email	String	Yes	The new users email address
InsertUserRequest	jobTitle	String	No	The new users job title
InsertUserRequest	companyName	String	No	The new users Company name

Output

Class	Name	Type	Required	Details
InsertUserResponse	success	boolean	Yes	True is succeeded False if failed
InsertUserResponse	returnCode	String	No	See Appendix for details
InsertUserResponse	returnMessage	String	No	Human readable message

Example

```
public static void testInsertUser(EVUserServiceStub stub) {
    try {
        // first we build the request...
        InsertUserDocument reqEnvelope = InsertUserDocument.Factory.newInstance();
        InsertUserRequest request = reqEnvelope.addNewInsertUser().addNewParam0();
        request.setNewUserPassword(ServiceClientHelper.generatedPassword);
        request.setNewUserId(ServiceClientHelper.generatedUserId);
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setEmail(ServiceClientHelper.generatedEmail);
        request.setFirstName(ServiceClientHelper.generatedFirstName);
        request.setLastName(ServiceClientHelper.generatedLastName);
        InsertUserResponseDocument resEnvelope = stub.insertUser(reqEnvelope);
        InsertUserResponse response = resEnvelope.getInsertUserResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                response.getReturnMessage());
            System.out.println(
              "user [" + ServiceClientHelper.generatedUserId + "] user has been
inserted.");
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                               response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

insertUserRole

This action creates a new user role and user mapping within ExtraView.

Input

Class	Name	Type	Required	Details
InsertUserRoleRequest	userId	String	Yes	The callers user name
InsertUserRoleRequest	password	String	Yes	The callers password
InsertUserRoleRequest	roleId	String	Yes	Therole id you want to add the user too
InsertUserRoleRequest	roleUserId	String	Yes	The user id you want to insert in the role

Output

Class	Name	Type	Required	Details
InsertUserRoleResponse	success	boolean	Yes	True is succeeded False if failed
InsertUserRoleResponse	returnCode	String	No	See Appendix for details
InsertUserRoleResponse	returnMessage	String	No	Human readable message

Example

```
public static void testInsertUserRole(EVUserServiceStub stub) {
    try {
        InsertUserRoleDocument reqEnvelope =
InsertUserRoleDocument.Factory.newInstance();
        InsertUserRoleRequest request =
reqEnvelope.addNewInsertUserRole().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setRoleUserId(ServiceClientHelper.generatedUserId);
        request.setRoleId(ServiceClientHelper.ADMIN_GROUP);
        InsertUserRoleResponseDocument resEnvelope = stub.insertUserRole(reqEnvelope);
        InsertUserRoleResponse response =
            resEnvelope.getInsertUserRoleResponse().getReturn();
        if (response.getSuccess()) {
```

```
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

itemExists

This function does a check to see if the item provided in itemId exists in ExtraView. It returns a simple Boolean result.

Input

Class	Name	Type	Required	Details
ItemExistsRequest	userId	String	Yes	The callers user name
ItemExistsRequest	password	String	Yes	The callers password
ItemExistsRequest	itemId	int	?	The int item id. This field is optional if itemIdStr is given, else it is required. It is recommended that you use itemIdStr
ItemExistsRequest	itemIdStr	String	?	The String item Id. This field is optional if itemId is used. It is recommended that you use itemIdStr

Output

Class	Name	Type	Required	Details
ItemExistsResponse	success	boolean	Yes	True is succeeded False if failed

ItemExistsResponse	returnCode	String	No	See Appendix for details
ItemExistsResponse	returnMessage	String	No	Human readable message
ItemExistsResponse	exists	boolean	Yes	The item in XML format

Example

```
public static void testItemExists(EVItemServiceStub stub) {
    try {
        ItemExistsDocument reqEnvelope = ItemExistsDocument.Factory.newInstance();
        ItemExistsRequest request = reqEnvelope.addNewItemExists().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setItemId(CREATED_ISSUE_ID);
        ItemExistsResponseDocument resEnvelope = stub.itemExists(reqEnvelope);
        ItemExistsResponse response = resEnvelope.getItemExistsResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

setCurrentUserRole

This action sets the role of the current user.

Input

Class	Name	Type	Required	Details
SetCurrentUserRoleRequest	userId	String	Yes	The callers user name
SetCurrentUserRoleRequest	password	String	Yes	The callers password

SetCurrentUserRoleRequest	roleId	String	Yes	The new current user role id

Output

Class	Name	Type	Required	Details
SetCurrentUserRoleResponse	success	boolean	Yes	True is succeeded False if failed
SetCurrentUserRoleResponse	returnCode	String	No	See Appendix for details
SetCurrentUserRoleResponse	returnMessage	String	No	Human readable message

Example

```
public static void testUpdateUserRole(EVUserServiceStub stub) {
    try {
        GetUserRolesResponse userRoleResponse = getUserRoles(stub);
        if (userRoleResponse.getSuccess()) {
            SetCurrentUserRoleDocument reqEnvelope =
                SetCurrentUserRoleDocument.Factory.newInstance();
            SetCurrentUserRoleRequest request =
                reqEnvelope.addNewSetCurrentUserRole().addNewRequest();
            request.setUserId(ServiceClientHelper.generatedUserId);
            request.setPassword(ServiceClientHelper.generatedPassword);
            request.setRoleId(userRoleResponse.getRolesArray()[0].getRoleId());
            SetCurrentUserRoleResponseDocument resEnvelope =
                stub.setCurrentUserRole(reqEnvelope);
            SetCurrentUserRoleResponse response =
                resEnvelope.getSetCurrentUserRoleResponse().getReturn();
            if (response.getSuccess()) {
                System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                    response.getReturnMessage());
            } else {
                System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                    response.getReturnMessage());
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

setWorkingProjectArea

This action sets the Business Area and Project for the current user within the database.

Input

Class	Name	Type	Required	Details
SetWorkingProjectAreaRequest	userId	String	Yes	The callers user name
SetWorkingProjectAreaRequest	password	String	Yes	The callers password
SetWorkingProjectAreaRequest	areaId	Integer	Yes	The area id to be set as the current area for the user
SetWorkingProjectAreaRequest	projectId	Integer	Yes	The project id to be set as the current project for the user

Output

Class	Name	Type	Required	Details
SetWorkingProjectAreaResponse	success	boolean	Yes	True is succeeded False if failed
SetWorkingProjectAreaResponse	returnCode	String	No	See Appendix for details
SetWorkingProjectAreaResponse	returnMessage	String	No	Human readable message

Example

```
public static void testSetWorkingProjectArea(EVSystemServiceStub stub) {
    try {
        SetWorkingProjectAreaDocument reqEnvelope =
            SetWorkingProjectAreaDocument.Factory.newInstance();
        SetWorkingProjectAreaRequest request =
            reqEnvelope.addNewSetWorkingProjectArea().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
```

```
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setAreaId(0);
        // GLOBAL AREA
        request.setProjectId(0);
        // MASTER PROJECT
        SetWorkingProjectAreaResponseDocument resEnvelope =
            stub.setWorkingProjectArea(reqEnvelope);
        SetWorkingProjectAreaResponse response =
            resEnvelope.getSetWorkingProjectAreaResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
            response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
            response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

updateItem

This API call updates an existing record in the ExtraView database. Except for the itemId, all fields are treated as optional. However, all defined business rules are executed and checked before and after the record is updated, therefore fields that are required or become required due to the rules must be provided.

Input

Class	Name	Type	Required	Details
UpdateItemRequest	userId	String	Yes	The callers user name
UpdateItemRequest	password	String	Yes	The callers password
UpdateItemRequest	itemId	int	?	The int item id. This field is optional if itemIdStr is given, else it is required. It is recommended that you use itemIdStr
UpdateItemRequest	itemIdStr	String	?	The String item Id. This field is optional if itemId is used. It is recommended that you use itemIdStr

UpdateItemRequest	sendEmail	boolean	Yes
UpdateItemRequest	itemFields	Array[]	Yes
ItemRecordField	fieldId	String	Yes
ItemRecordField	fieldTitle	String	Yes
ItemRecordField	fieldValue	String	No
ItemRecordField	row	Integer	No

Optional parameters for handling document and image files:

Class	Name	Type	Required	Details
ItemFileField	itemImages	Array[]	No	The array of image files and data
ItemFileField	itemDocuments	Array[]	No	The array of document files and data
ItemFileField	ddName	String	No	The data dictionary name of the file
ItemFileField	fileName	String	No	The target file name to be saved
ItemFileField	file	String	No	The file
ItemFileField	description	String	No	The description of the file
ItemFileField	charset	String	No	The charset of the file

ItemFileField	contentType	String	No	The content type - e.g. "image/GIF"
ItemFileField	rowId			The repeating row Id
ItemFileField	rowNumber	int	No	The row number - currently not supported

Output

Class	Name	Type	Required	Details
UpdateItemResponse	success	boolean	Yes	True is succeeded False if failed
UpdateItemResponse	returnCode	String	No	See Appendix for details
UpdateItemResponse	returnMessage	String	No	Human readable message
UpdateItemResponse	itemId	int	Yes	The int value of the ID of the inserted item
UpdateItemResponse	itemIdStr	String	Yes	The recommended field to obtain the ID of the inserted item

Example

```
public static void testUpdateItem(EVItemServiceStub stub) {
    try {
        UpdateItemDocument reqEnvelope = UpdateItemDocument.Factory.newInstance();
        UpdateItemRequest request = reqEnvelope.addNewUpdateItem().addNewRequest();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        List items = new ArrayList();
        ItemRecordField item = null;
        for (int i = 0; i < 4; i++) {
            item = ItemRecordField.Factory.newInstance();
            item.setRow(0);
            // no repeating rows for this example
```

```
            switch (i) {
                case 0:
                    item.setFieldId("status");
                    item.setFieldValue("NEW");
                    break;
                case 1:
                    item.setFieldId("assigned_to");
                    item.setFieldValue(ServiceClientHelper.generatedUserId);
                    break;
                case 2:
                    item.setFieldId("short_description");
                    item.setFieldValue("This is the short desc or title");
                    break;
                case 3:
                    item.setFieldId("description");
                    item.setFieldValue("this is the long desc");
                    break;
                default:
                    item.setFieldId("unknown");
                    item.setFieldValue("");
                    break;
            }
            items.add(item);
        }
        request.setItemFieldsArray(items.toArray(new ItemRecordField[items.size()]));
        request.setSendEmail(false);
        UpdateItemResponseDocument resEnvelope = stub.updateItem(reqEnvelope);
        UpdateItemResponse response = resEnvelope.getUpdateItemResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

Example 2 - Handling Document & Image Fields

```
public static void testInsertItem(EVItemServiceStub stub) {
    try {
        File imageFile = new File("./test.GIF");
        DataInputStream dis = new DataInputStream(new FileInputStream(imageFile));
        BufferedReader br = new BufferedReader(new InputStreamReader(dis));
        List bFileList = new ArrayList();
        try {
            byte b;
            while (true) {
                b = dis.readByte();
                bFileList.add(b);
            }
        } catch (Exception e) { /* handle the EOF the lazy way. */
        }
        byte[] fileBytes = new byte[bFileList.size()];
        for (int i = 0; i items = new ArrayList(); ItemRecordField item = null;
        for (int i = 0; i < 4; i++) {
            item = ItemRecordField.Factory.newInstance();
```

```
            item.setRow(0);
            // no repeating rows for this example
            switch (i) {
                case 0:
                    item.setFieldId("status");
                    item.setFieldValue("NEW");
                    break;
                case 1:
                    item.setFieldId("assigned_to");
                    item.setFieldValue(ServiceClientHelper.generatedUserId);
                    break;
                case 2:
                    item.setFieldId("SHORT_DESCR");
                    item.setFieldValue("This is the short desc or title");
                    break;
                case 3:
                    item.setFieldId("DESCRIPTION");
                    item.setFieldValue("this is the long desc");
                    break;
                default:
                    item.setFieldId("unknown");
                    item.setFieldValue("");
                    break;
            }
            items.add(item);
        }
        request.setItemFieldsArray(items.toArray(new ItemRecordField[items.size()]));
        request.setSendEmail(false);
        List itemDocumentFields = new ArrayList();
        ItemFileField document = ItemFileField.Factory.newInstance();
        document.setCharset("UTF-8");
        document.setContentType("image/gif");
        document.setDescription("This is a test image file in a document field!");
        document.setFile(fileBytes); document.setDdName("DOCUMENT");
        document.setFileName(imageFile.getName());
        itemDocumentFields.add(document);
        request.setItemDocumentsArray(itemDocumentFields.toArray(new
            ItemFileField[itemDocumentFields.size()]));
        InsertItemResponseDocument resEnvelope = stub.insertItem(reqEnvelope);
        InsertItemResponse response = resEnvelope.getInsertItemResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                            response.getReturnMessage());
            System.out.println("Item [" + response.getItemId() + "] was created.");
            CREATED_Item_ID = response.getItemId();
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                            response.getReturnMessage());
        }
        } catch (Exception e) {
            e.printStackTrace();
            System.err.println("\n\n\n");
        }
    }
    public static void testUpdateItem(EVItemServiceStub stub) {
        try {
            File imageFile = new File("./test.GIF");
            DataInputStream dis = new DataInputStream(new FileInputStream(imageFile));
            BufferedReader br = new BufferedReader(new InputStreamReader(dis));
            List bFileList = new ArrayList();
            try {
                byte b;
```

```
                while (true) {
                    b = dis.readByte();
                    bFileList.add(b);
                }
            } catch (Exception e) { /* handle the EOF the lazy way. */
            }
            byte[] fileBytes = new byte[bFileList.size()];
            for (int i = 0; i
                items = new ArrayList();
                ItemRecordField item = ItemRecordField.Factory.newInstance();
                item.setRow(0);
                // no repeating rows for this example
                item.setFieldId("SHORT_DESCR");
                item.setFieldValue("This is the short desc or title -- changed!");
                items.add(item);
                request.setItemFieldsArray(items.toArray(new
ItemRecordField[items.size()]));
                request.setSendEmail(false);
                List itemImageFields = new ArrayList();
                ItemFileField image = ItemFileField.Factory.newInstance();
                image.setCharset("UTF-8");
                image.setContentType("image/gif");
                image.setDescription("This is a test image file in a image field!");
                image.setFile(fileBytes);
                image.setDdName("IMAGE");
                image.setFileName(imageFile.getName());
                itemImageFields.add(image);
                request.setItemImagesArray(itemImageFields.toArray(new
                    ItemFileField[itemImageFields.size()]));
                UpdateItemResponseDocument resEnvelope = stub.updateItem(reqEnvelope);
                UpdateItemResponse response =
                    resEnvelope.getUpdateItemResponse().getReturn();
                if (response.getSuccess()) {
                    System.out.println("success: [" + response.getReturnCode() + "] : "
+
                                                    response.getReturnMessage());
                System.out.println("Item [" + response.getItemId() + "] was changed.");
                CREATED_Item_ID = response.getItemId();
            } else {
                System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
            }
            } catch (Exception e) {
                e.printStackTrace();
                System.err.println("\n\n\n");
            }
        }
```

updateUser

This action allows the user to update the attributes of an existing user within ExtraView.

Input

Class	Name	Type	Required	Details

UpdateUserRequest	userId	String	Yes	The callers user name
UpdateUserRequest	password	String	Yes	The callers password
UpdateUserRequest	updateUserId	String	Yes	The desired new user id (must be unique)
UpdateUserRequest	newPassword	String	No	The desired new password
UpdateUserRequest	oldPassword	String	No	The users old password
UpdateUserRequest	firstName	String	No	The users first name
UpdateUserRequest	lastName	String	No	The users last name
UpdateUserRequest	email	String	No	The users email address
UpdateUserRequest	jobTitle	String	No	The users job title
UpdateUserRequest	companyName	String	No	The users Company name
UpdateUserRequest	addressLine1	String	No	The users address (line 1)
UpdateUserRequest	addressLine2	String	No	The users address (line 2)
UpdateUserRequest	city	String	No	The users city
UpdateUserRequest	state	String	No	The users state

UpdateUserRequest	postalCode	String	No	The users postal code
UpdateUserRequest	country	String	No	The users country
UpdateUserRequest	workPhoneNumber	String	No	The users work phone number
UpdateUserRequest	homePhoneNumber	String	No	The users home phone number
UpdateUserRequest	cellPhoneNumber	String	No	The users cell phone number
UpdateUserRequest	faxNumber	String	No	The users fax phone number
UpdateUserRequest	pagerNumber	String	No	The users pager phone number

Output

Class	Name	Type	Required	Details
UpdateUserResponse	success	boolean	Yes	True is succeeded False if failed
UpdateUserResponse	returnCode	String	No	See Appendix for details
UpdateUserResponse	returnMessage	String	No	Human readable message

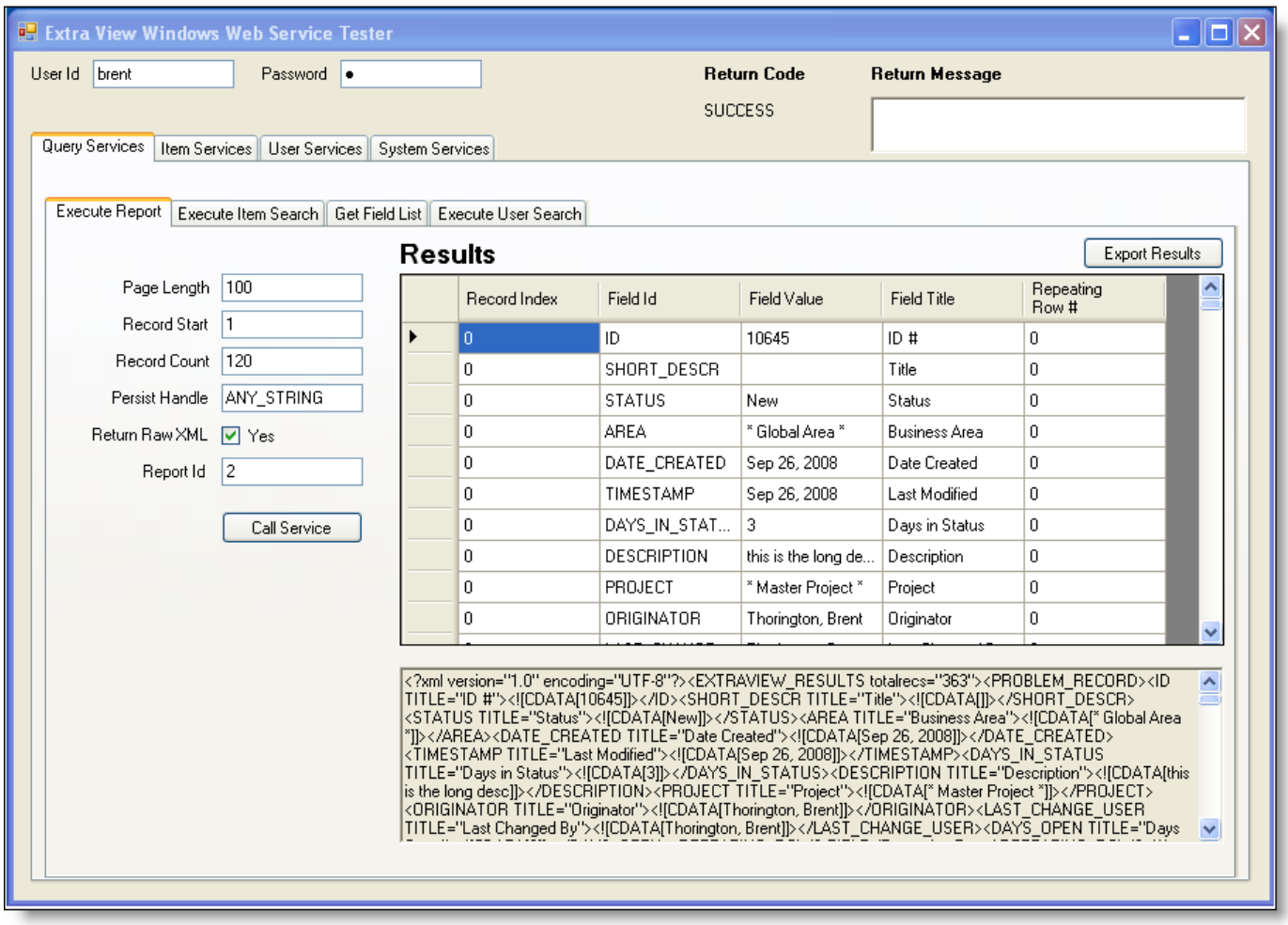
Example

```
public static void testUpdateUser(EVUserServiceStub stub) {
    try {
        UpdateUserDocument reqEnvelope = UpdateUserDocument.Factory.newInstance();
        UpdateUserRequest request = reqEnvelope.addNewUpdateUser().addNewParam0();
        request.setUserId(ServiceClientHelper.ADMIN_USER_ID);
        request.setPassword(ServiceClientHelper.ADMIN_PASSWORD);
        request.setUpdateUserId(ServiceClientHelper.generatedUserId);
        request.setOldPassword(ServiceClientHelper.generatedPassword);
```

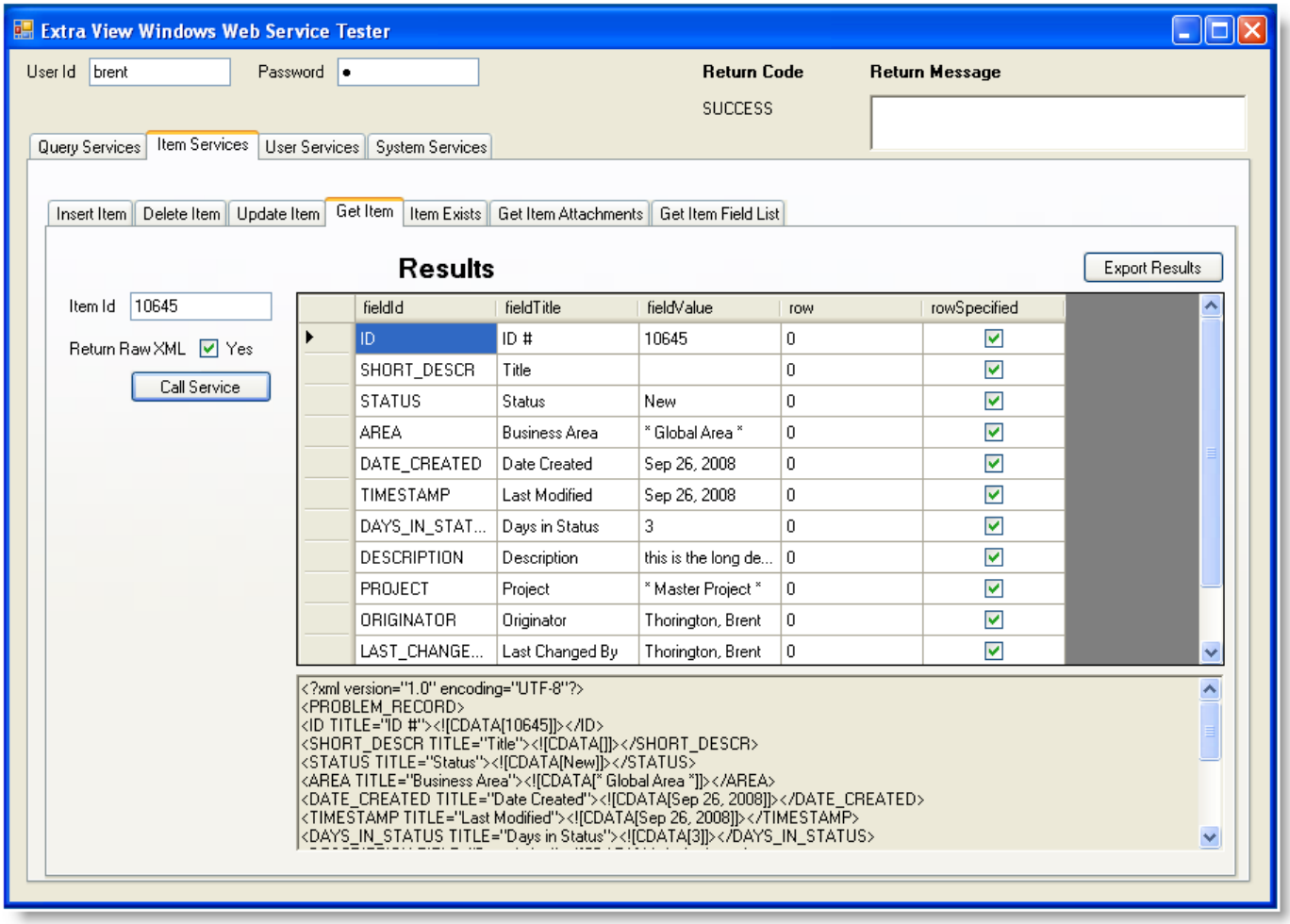
```
        request.setNewPassword("UP" + ServiceClientHelper.generatedPassword);
        UpdateUserResponseDocument resEnvelope = stub.updateUser(reqEnvelope);
        UpdateUserResponse response = resEnvelope.getUpdateUserResponse().getReturn();
        if (response.getSuccess()) {
            System.out.println("success: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
            System.out.println
              ("user [" + ServiceClientHelper.generatedUserId + "] user has been
inserted.");
        } else {
            System.out.println("failure: [" + response.getReturnCode() + "] : " +
                                                response.getReturnMessage());
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.err.println("nnn");
    }
}
```

Sample .Net Client

A full sample client program, developed in Microsoft Visual Studio 8, is provided as part of the implementation. The full source code is available. Every action described in this guide is implemented within the client program. The program has a single configuration file. As an example, this configuration file provides the path to the ExtraView server.



Sample interface utilizing ExtraView Web Services



Example service to retrieve an item from the database displaying the results

Appendix

- Add new comment

Return Codes

Return Code	Definition
SUCCESS	The web service successfully ran without errors or warnings
FAILED	The web service failed to run successfully
MULTI_FAILED	Multiple failures occurred (for batch runs)
WARNING	The web service got unusual results from ExtraView and may or may not have ran successfully
UNKNOWN	The web service was unable to determine the correct status to be returned for the run. Generally, this should be treated the same as FAILED run