Enable API Overview

For

Version 9

EnterWorks®, Inc.
46040 Center Oak Plaza Suite 115
Sterling, VA 20166

©EnterWorks, Inc.
Loudoun Tech Center
46040 Center Oak Plaza
Suite 115
Sterling, VA 20166

1.888.242.8356 (Sales and General Information)
1.888.225.2705 (U.S. Support)
http://www.enterworks.com

EnterWorks ® Enable PIM™ *User's Manual*
Version 9

Contents

1 Document Conventions

This EnterWorks document uses the following typographic conventions:

Convention	Usage
pathnames	Pathnames are shown with backslashes, as for Windows systems.
`Courier New font`	Denotes sample code, for example, Java, IDL, and command line information. May be used to denote filenames and pathnames, calculations, code samples, registry keys, path and file names, URLs, messages displayed on the screen. If *italicized* and in angle brackets (< >), it denotes a variable.
Calibri Font (bold)	When used in body text, it denotes an object, area, list item, button, or menu option within the graphical user interface; or a database name or database-related object. (Examples: the **Save** button; the **Product** tab; the **Name** field; the **SKU** repository) Can also be used to denote text that is typed in a text box. (Example: Type "**trackingNo**" in the Name field)
Blue underlined text	Words, phrases or numbers in blue are active links that can be clicked. Clicking these active links will bring the user to the required information, steps, pages chapters, or URL.

2 Customer Support

EnterWorks provides a full spectrum of customer support. Check your maintenance contract for details about the level of support purchased. The first time you contact customer support, you will be issued a customer identification number. Keep this number for future reference when using the EnterWorks customer support service.

How to reach us	Comments
On the Web: http://support.enterworks.com Via email: support@enterworks.com	For detailed discussions of hardware, software, configuration issues, or Helpdesk credentials, contact your EnterWorks representative.
Phone: U. S. Support 1. 888. 225. 2705	Support hours are from 9:00 a.m. to 8:00 p.m., United States Eastern Time (-0500 UTC), Monday through Friday.
Postal mail: EnterWorks Acquisition Inc. Customer Support Team Loudoun Tech Center 46040 Center Oak Plaza Suite 115 Sterling, VA 20166 USA	Please include your telephone number and customer identification number or project name in your letter.

3 Enable API Services

The following sections describe what are included in the Enable Web Services API and how to use them.

The Enable API Services provides an extension to allow third party applications, not just limited to Java based applications, to integrate with the Enable system. It provides most of the runtime support for the PIM management functionality. However, it is not an alternative to the web browser based user interface because it lacks most of the administrative capabilities such as user, security, repository mapping management etc.

The Enterworks Product Information Manager architecture is based on the Java 2 Enterprise Edition (J2EE) platform, and as such makes use of several J2EE platform technologies such as Enterprise Java Beans (EJBs), Java Servlets, and Java Messaging Service (JMS). Custom applications developed for Enable must be written in Java, but do not necessarily require knowledge of the internal usage of J2EE technologies employed by Enable. The Enable public API available to custom applications are written entirely in Java and do not require JMS programming skills or knowledge of EJBs or Java Servlets. These Java APIs are actually libraries of compiled codes that can be used in Enable.

4 SOAP Services

The interface provides the venue by which independent systems communicate with each other. The Enable interface is programming language independent and Operating System independent.

The Enable Web Services APIs are described in the pim.wsdl (Web Service Description Language). The Enable installer already bundles the necessary java class files during the installation; thus, there is no need to re-generate the java client from the WSDL file. However, this is quite acceptable to do so. See the “How to Use” section for details on how to use the pre-bundled java classes. Non-Java applications can use the WSDL file as the basis to generate their corresponding client access commands in their native languages.

The core interface in the Enable Web Services API is called “Server”. Others are simply supporting data types that are used to either support the SOAP protocol or used as information holders.

The supporting data types are as follows:

- AttrCodeSetMapping
- AttrData
- Attribute
- CodeSet
- CodeSetDetail

- EJBServer
- EnableTreeNode
- Group
- Item
- ItemHistory
- ItemHistoryDiff
- ItemList
- ItemState
- KeyValuePair
- Media
- MediaGroup
- MediaReference
- Package-frame
- Package-summary
- Package-tree
- Repository
- RepositoryGroup
- RepositoryViewMapping
- SearchCondition
- Server
- ServerSoapBindingImpl
- Session
- Subscription
- TextSearchConfig
- TextSearchConfigAttr
- TradingPartner
- TransactionStatus
- UserPreference

4.1 Services

The Enable Web Services API are divided into three major functional groups: metadata retrieval, repository/item management and tools.

4.1.1 Metadata retrieval

- createSubscription - Creates a subscription in the server by a given Subscription object.
- deleteSubscription - Deletes a subscription object from the database.

- exportRepository - Perform export for a particular repository with specified userpreference.
- getCodeSet - Retrieves the metadata information for the code set.
- getCodeSetList - Retrieves a list of code sets the user can access.
- getCodeSetDetail - Retrieves the code set detail including code, code description, parent code, and other information.
- getLowerLevelCodeSetDetail - Retrieves the lower level code set detail for a particular code set and a code.
- getCustomerList - Retrieves a list of customers. The return list construction depends on the user's access rights.
- getCustomerListBySupplierGLN - Retrieves a list of customers for a particular supplier.
- getSupplierList - Retrieves a list of suppliers. The return list construction depends on the user's access rights.
- getGroupList - Retrieves the list of groups the current user belongs to.
- getRepositoryByName - Retrieve the repository by name.
- getRepositoryList - Retrieves the list of repositories the current user can access.
- getRepositoryAttributeList – Retrieves the attribute list for a particular repository.
- getUserPreferenceList - Retrieves the user preferences the current user can access.
- getUserPreferenceAttributeList - Retrieve attributes for a particular user preference.
- getJobStatus - Retrieves the job status.
- getCodeSetDetailByCode - Retrieve code set detail including code, code description and parent code etc.
- getCodeSetDetailByName - Retrieve code set detail including code, code description and parent code etc.
- getAttrCodeSetMapping – Retrieves the attribute and code set mapping for a particular repository.
- getCustomerList - Retrieves a list of customers.
- getCustomerListBySupplierGLN - Retrieves a list of customers for a particular supplier.
- getEnableTree - Retrieves enAble tree nodes.
- getItemHistory - Returns a single ItemHistory object based on its itemHistoryId (obtained from `getRepositoryItemHistoryList()`).
- getItemHistoryAttrDiffList - Returns a list of ItemHistoryDiff objects for two ItemHistory objects.
- getItemHistoryList - Returns an array of ItemHistory objects for the specified repositoryId and itemId based on the date and userLogin search parameters.

- getLowerLevelCodeSetDetailByName - Retrieve lower level code set detail for a particular code set and a code.
- getMediaContent - Retrieves media content for a particular media object identified by media ID.
- getMediaGroupList – Retrieves Media Groups.
- getMediaReferenceList - Retrieves references for a particular media object identified by media ID.
- getRepositoryGroupList - Retrieves the repository group list.
- getRepositoryListByGroupName - Retrieves a list of repository objects for a particular repository group.
- getSubscriptionList - To retrieve an array of Subscription objects.
- getSupplierList - Retrieves a list of suppliers.
- getTextSearchConfigList - Retrieves a list of TextSearchConfig objects for a particular repository.
- getUserList - Retrieve the list of users the current user has access to see.
- getUserPreferenceByName - Retrieve user preference by name.
- getViewMappingList - Retrieves an array of RepositoryViewMapping objects for a repository.
- subscribeSubscription - To subscribe the subscription.
- searchCodeSetDetail - Search the codeSet detail using search string and search method and return an array of CodeSetDetail Objects.
- setDataPoolSyncOutStatus - Update sync out histrory status based on message data pool response message identifier.
- setDataPoolSyncOutStatusByFile - Update sync out histrory status based on data pool response message file name.
- syncInRepository - Perform sync in operation for a particular repository.
- unsubscribeSubscription - To remove the subscription.
- updateSubscription - Updates a subscription object with the given Subscription object.
- updateSubscriptionState - Updates the subscription state of a given subscription object.
- validateRepositoryOption - Perform validation for a particular repository with specified optionCode.

4.1.2 Repository/Item Management

- abortJob - Abort a job with the given jobId.
- createMedia - Creates the media object.
- createMediaByURL - Creates media object using URL.

- deleteItem - Deletes a particular item.
- deleteMedia - Deletes a media object identified by media ID.getItem - Retrieves the item detail.
- getItemState - Retrieves the item state.
- createItem - Creates a new item for a particular master repository.
- delistItem - Mark item delist. DeleteItem - Deletes a particular item.
- updateItem - Updates the item data
- updateItems - Updates items with search conditions.
- searchItem - Performs item search based on the search conditions for a given repository.
- searchItemByConfig - Performs item search based on their configurations.
- searchItemByKeyword - Performs item search using keywords.
- searchItemByRepository - You can search items in a particular repository based on simple search conditions such as "AND" or "OR" joined.
- searchItemByUserPreference - You can search items for a particular user preference.
- searchMedia - Performs media object search.
- setItemState - Updates the item state.
- setDataPoolResponseStatus – Updates the item status based on data pool response message identifier.
- setDataPoolReponseStatusByFile – Updates the item status based on data pool response message file name.
- setDataPoolResponseStatusByPrimKey - Update item status based on data pool response message primary key.
- setDataPoolReponseStatusByTransaction - Updates the item status based on data pool response message identifier and transactions.
- validateRepository - Performs validation for a particular repository.
- syncOutRepository - Performs the sync out operation for a particular repository.
- createCodeSetDetail - Creates a code set detail record for a particular code set.
- deleteCodeSetDetail - Delete a code detail record for a particular code set.
- isValidParent - Will interrogate the hasErrorInd of the parent item and all of its linked children with the given link relationship and return the results.
- updateCodeSetDetail - Updates a code set detail record for a particular code set.
- updateItemsByKeywordSearch - Update items using key word using text search.
- updateItemTriggerOption - Updates item data and status.
- updateMedia - Updates the media object.
- updateMediaByURL – Updates the media object by URL.
- updateMediaContent - Updates media content only without the associated meta data.

- updateMediaContentByURL - Updates media content only, without the associated metadata, using content URL.
- updateMediaMetaData - Updates the media metadata only and not the content.
- getParentRepositoryLinkList - Retrieves an array of RepositoryLink objects for a parent repository.
- getChildRepositoryLinkList - Retrieves an array of RepositoryLink objects for a child repository.

4.1.3 Tools

- Login - Logs into e-PIM server to obtain a valid session.
- Logout - Logs out from e-PIM server and terminates user session.
- ChangeUserPassword - Changes the password for the login user.
- loginByGID - Log into Enable server using global ID.

The method by which these available Web service functions can be invoked is provided in the Enable Integrated Web Services Javadoc, which provides the necessary documentation for Enable Web Services. The Web Services Javadoc is the generated zip file containing the API documentation of Enable's automated resources accessed via the Internet. It is extracted from the comments in the Java source code and made available in HTML format.

For more information on Java Web Services methods and calls, particularly metadata retrieval, repository/item management and tools, refer to the Enable Web Services Javadoc. To access the Javadoc, navigate to the directory below and locate the wsdoc file:

```
C:\Enterworks\Enable\documents
```

4.2 SOAP Service Sample Code

Java source codes are compiled into Java Class files that can be loaded to execute Methods needed to perform the different functions and capabilities of the Enable application, particularly the Web Services component. Presented below is the sample code or script that executes some Enable methods described above.

The sample below shows how to perform sync in, sync out, and a validation method call using the Enable 7 Web Services API:

```
package com.enterworks.Enable.test.webservice;

import java.util.ArrayList;
import com.enterworks.Enable.test.webservice.KeyValuePair;
import com.enterworks.Enable.test.webservice.Repository;
```

```
/* Demonstrate sync in, sync out and validation method calls using Enable Web
services API
 */
public class PimWSClientSample {
 private static String hostURL =
  "http://sulley:8888/webcm/services/server";

 private static String usage = "Usage: java PimWSClient syncout|validatation
<repository name>";

 public static void main(String[] args) {
  com.enterworks.Enable.test.webservice.ServerSoapBindingStub binding = null;
  if (args.length < 2) {
   System.err.println(usage);
   return;
  }
  if (args[0].equalsIgnoreCase("syncout")) {
   System.out.println("Sync out repository:" + args[1]);
  } else if (args[0].equalsIgnoreCase("validation")) {
   System.out.println("Validation repository:" + args[1]);
  } else {
   System.err.println(usage);
   return;
  }
  try {
   // login first

   com.enterworks.Enable.test.webservice.ServerServiceLocator loc = new
    com.enterworks.Enable.test.webservice.ServerServiceLocator();
   binding = (com.enterworks.Enable.test.webservice.ServerSoapBindingStub)
    loc.getserver(new java.net.URL(hostURL));
   com.enterworks.Enable.test.webservice.Session mySession = login(binding);

   // validate result

   System.err.println("login session:" +
    mySession.getSessionId());
   System.err.println("login user:" + mySession.getUserId());

   // retrieve repository list

   Repository[] repList = null;
   Repository rep = null;
   repList = binding.getRepositoryList(mySession);

   if (repList == null) {
    System.out.println("No repository to retrieve.");
    return;
   }
   for (int i = 0; i < repList.length; i++) {
    if (args[1].equalsIgnoreCase(repList[i].getName())) {
     System.out.println("Find matching repository:" + args[1]);
     rep = repList[i];
```

```
      break;
     }
    }
    if (rep == null) {
     System.err.println("No matching repository found.");
     return;
    }
    if (args[0].equalsIgnoreCase("syncout")) {
     binding.syncOutRepository(mySession, rep, false);
    } else if (args[0].equalsIgnoreCase("validation")) {
     binding.validateRepository(mySession, rep);
    } else {
     System.err.println(usage);
     return;
    }

    //logout

    binding.logout(mySession);
    } catch (javax.xml.rpc.ServiceException jre) {
    if (jre.getLinkedCause() != null) jre.getLinkedCause().printStackTrace();
    throw new junit.framework.AssertionFailedError("JAX-RPC
     ServiceException caught: " + jre);
    } catch (Exception e) {
     System.err.println(e.getMessage());
    }
   }
   /*
    * common login wrapper routine.
    */
  public static com.enterworks.Enable.test.webservice.Session
login(com.enterworks.Enable.test.webservice.ServerSoapBindingStub binding) {
    try {
     com.enterworks.Enable.test.webservice.Session value = null;
     value = binding.login("system", "system", true, null);
     return value;
    } catch (Exception e) {
     System.out.println("Can't login." + e.getMessage());
    }
    return null;
   }
   /* login */
   /*
    *    Update the item attribute values.
    *    Please refer to the java doc for API description.
    */
  public static void updateItem() {
    com.enterworks.Enable.test.webservice.ServerSoapBindingStub binding =
null;
    try {
     // login first

     com.enterworks.Enable.test.webservice.ServerServiceLocator loc = new
```

```
    com.enterworks.Enable.test.webservice.ServerServiceLocator();
    binding = (com.enterworks.Enable.test.webservice.ServerSoapBindingStub)
     loc.getserver(new java.net.URL(hostURL));
    com.enterworks.Enable.test.webservice.Session mySession = login(binding);

    // validate result

    System.err.println("login session:" +
     mySession.getSessionId());
    System.err.println("login user:" + mySession.getUserId());
    java.lang.String value = null;

    // should do repository retrieval instead of doing the following
    // this code assumes the repository is known.

    Repository rep = new Repository();
    rep.setRepositoryId(1);
    rep.setMasterIndicator(true);
    ArrayList l = new ArrayList();

    // again, should retrieve attribute list instead of hard code.

    KeyValuePair kvp1 = new KeyValuePair("F_11", "123");
    KeyValuePair kvp2 = new KeyValuePair("F_22", "abc");
    l.add(kvp1);
    l.add(kvp2);
    binding.updateItem(login(binding), rep, 1, (KeyValuePair[]) l.toArray(new
         KeyValuePair[l.size()]));
    System.out.println("done.");
   } catch (Exception e) {
    System.err.println(e.getMessage());
   }
  }
  /*
   *    Update item data pool response.
   *    Please read the java doc for API description.
   */
  public static void updateItemStatus() {
   com.enterworks.Enable.test.webservice.ServerSoapBindingStub binding =
    null;

   try {

    // login first

    com.enterworks.Enable.test.webservice.ServerServiceLocator loc = new
com.enterworks.Enable.test.webservice.ServerServiceLocator();
    binding = (com.enterworks.Enable.test.webservice.ServerSoapBindingStub)
loc.getserver(new java.net.URL(hostURL));
    com.enterworks.Enable.test.webservice.Session mySession = login(binding);

    // validate result
```

```
    System.err.println("login session:" +
     mySession.getSessionId());
    System.err.println("login user:" + mySession.getUserId());
    java.lang.String value = null;

    // assume the message id is "UCCNET_RESPONSE1"
    // read java doc to see the actual status code meaning.

    binding.setDataPoolResponseStatus(login(binding),
     "UCCNET_RESPONSE1", 0, "Sync from UCCNet Ok.");
   } catch (Exception e) {
    System.err.println(e.getMessage());
   }
  }
  public static void updateItemResponse() {
   com.enterworks.Enable.test.webservice.ServerSoapBindingStub binding =
null;

   try {

    // login first

    com.enterworks.Enable.test.webservice.ServerServiceLocator loc = new
com.enterworks.Enable.test.webservice.ServerServiceLocator();
    binding = (com.enterworks.Enable.test.webservice.ServerSoapBindingStub)
loc.getserver(new java.net.URL(hostURL));
    com.enterworks.Enable.test.webservice.Session mySession = login(binding);

    // validate result

    System.err.println("login session:" +
     mySession.getSessionId());
    System.err.println("login user:" + mySession.getUserId());

    // should retrieve repository list first
    // this code assume it exists.

    Repository rep = new Repository();
    rep.setRepositoryId(1010946);
    rep.setMasterIndicator(true);

    // see java doc for actual status code meaning

    binding.setItemState(login(binding), rep, 1011019, 2, true, 3, true, 1,
false, 0, true);
   } catch (Exception e) {
    System.err.println(e.getMessage());
   }
  }
 }
```

5 Rest Services

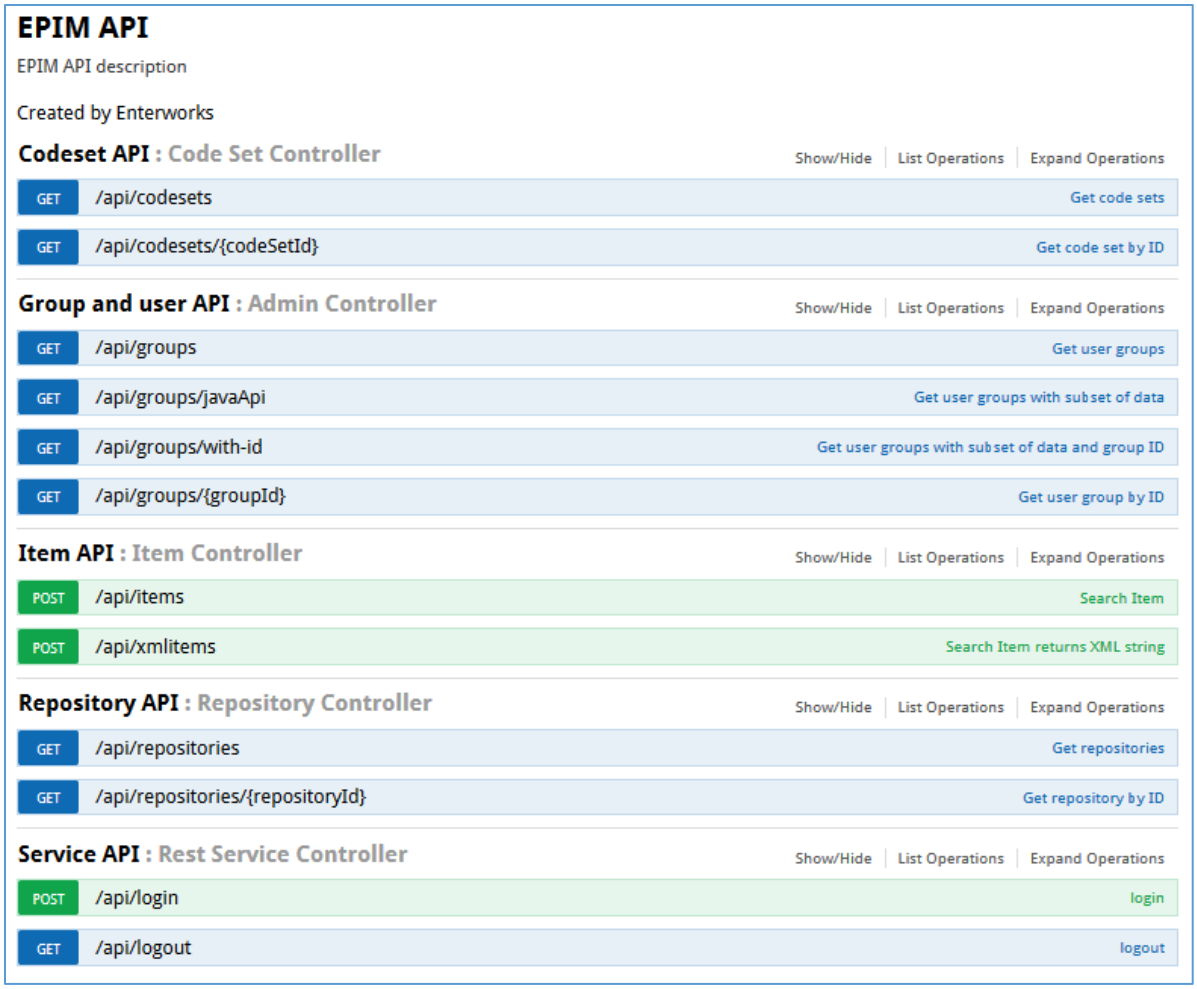
With the representational state transfer (REST) style architecture, requests and responses are built around the transfer of representations of resources. Resources are identified by global IDs that typically use a uniform resource identifier (URI). Client applications use HTTP methods (such as GET, POST, PUT, or DELETE) to manipulate the resource or collection of resources. Refer to the SOAP API Services above for additional information on available service details

Generally, a GET method is used to get or list the resource or collection of resources, POST is used to create, PUT is used to update or replace, and DELETE is for removing the resource. For example, GET http://host/context/user/12345 gets the representation of the user with the ID 12345. The response representation could be an XML or JSON that contains the detailed user information.

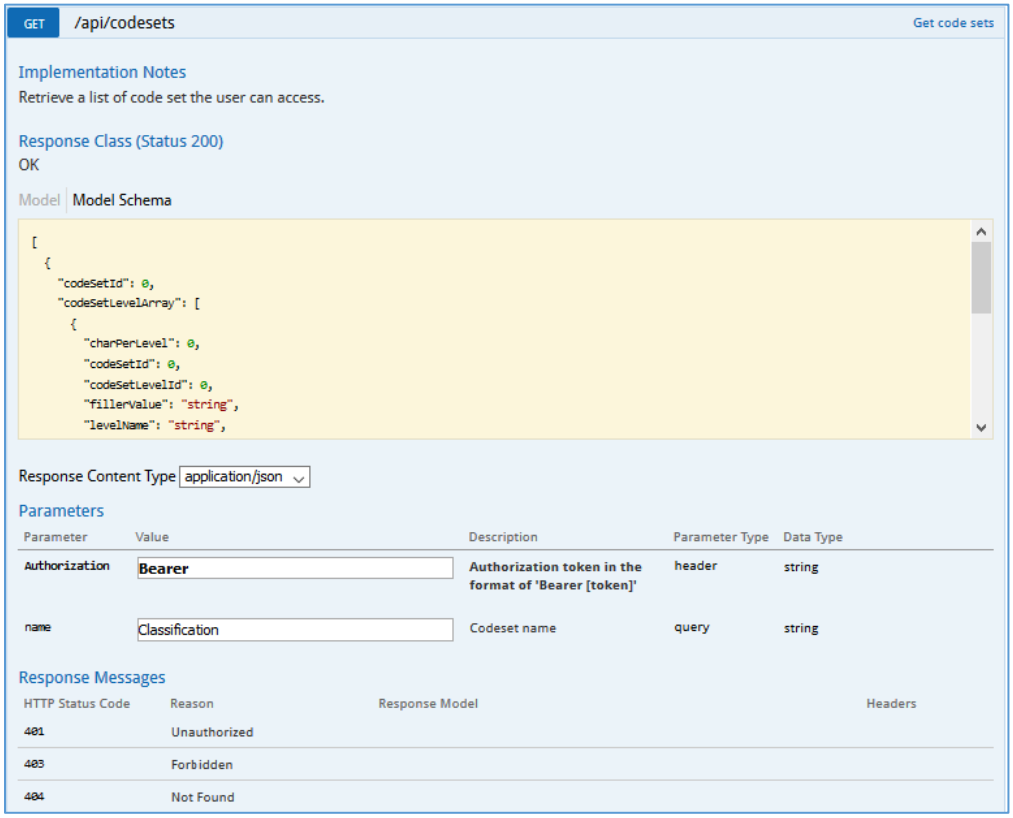
Spring 4 (with Oauth2 support) is the framework for the Enable RESTful Web Service layer.

5.1 Rest Service APIs

Enterworks is currently developing Restful Web Services for use with the Enable system. The following APIs are available.


EPIM API
EPIM API description
Created by Enterworks
Codeset API : Code Set Controller
Show/Hide | List Operations | Expand Operations
GET /api/codesets Get code sets
GET /api/codesets/{codeSetId} Get code set by ID
Group and user API : Admin Controller
Show/Hide | List Operations | Expand Operations
GET /api/groups Get user groups
GET /api/groups/javaApi Get user groups with subset of data
GET /api/groups/with-id Get user groups with subset of data and group ID
GET /api/groups/{groupId} Get user group by ID
Item API : Item Controller
Show/Hide | List Operations | Expand Operations
POST /api/items Search Item
POST /api/xmlitems Search Item returns XML string
Repository API : Repository Controller
Show/Hide | List Operations | Expand Operations
GET /api/repositories Get repositories
GET /api/repositories/{repositoryId} Get repository by ID
Service API : Rest Service Controller
Show/Hide | List Operations | Expand Operations
POST /api/login login
GET /api/logout logout

5.1.1 Code Sets



GET /api/codesets — Get code sets

Implementation Notes

Retrieve a list of code set the user can access.

Response Class (Status 200)

OK

Model | Model Schema

```
[
  {
    "codeSetId": 0,
    "codeSetLevelArray": [
      {
        "charPerLevel": 0,
        "codeSetId": 0,
        "codeSetLevelId": 0,
        "fillerValue": "string",
        "levelName": "string",
```

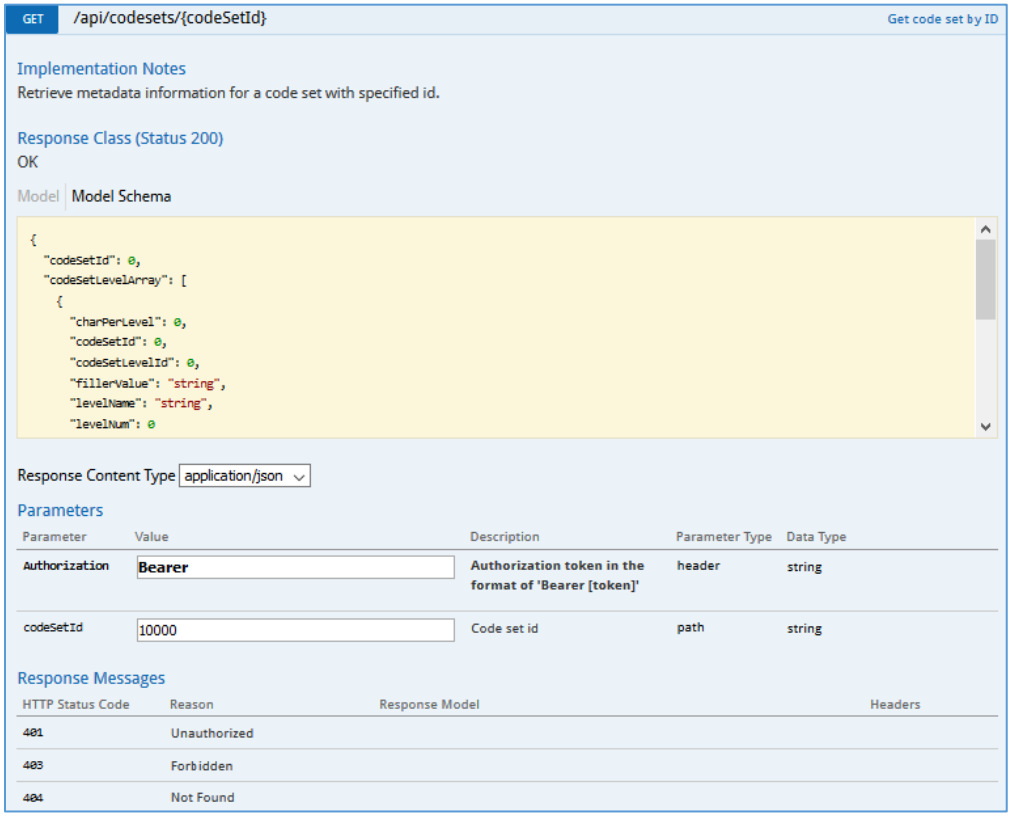
Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	Bearer	Authorization token in the format of 'Bearer [token]'	header	string
name	Classification	Codeset name	query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		



GET /api/codesets/{codeSetId} — Get code set by ID

Implementation Notes

Retrieve metadata information for a code set with specified id.

Response Class (Status 200)

OK

Model | Model Schema

```
{
  "codeSetId": 0,
  "codeSetLevelArray": [
    {
      "charPerLevel": 0,
      "codeSetId": 0,
      "codeSetLevelId": 0,
      "fillerValue": "string",
      "levelName": "string",
      "levelNum": 0
```

Response Content Type: application/json

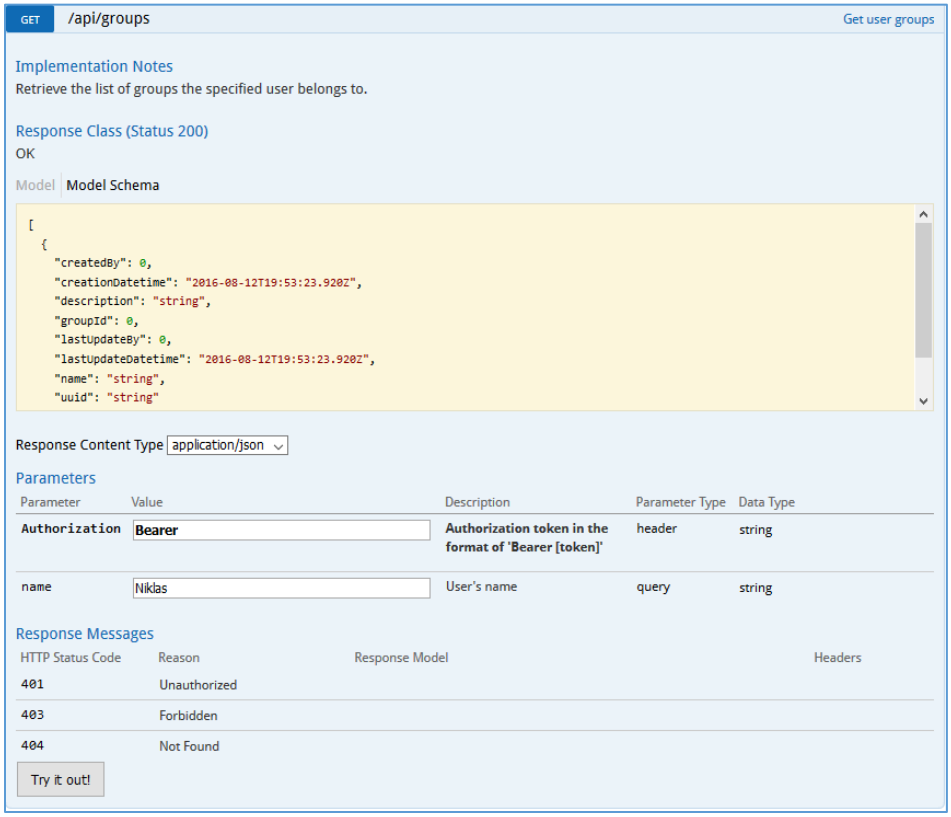
Parameters

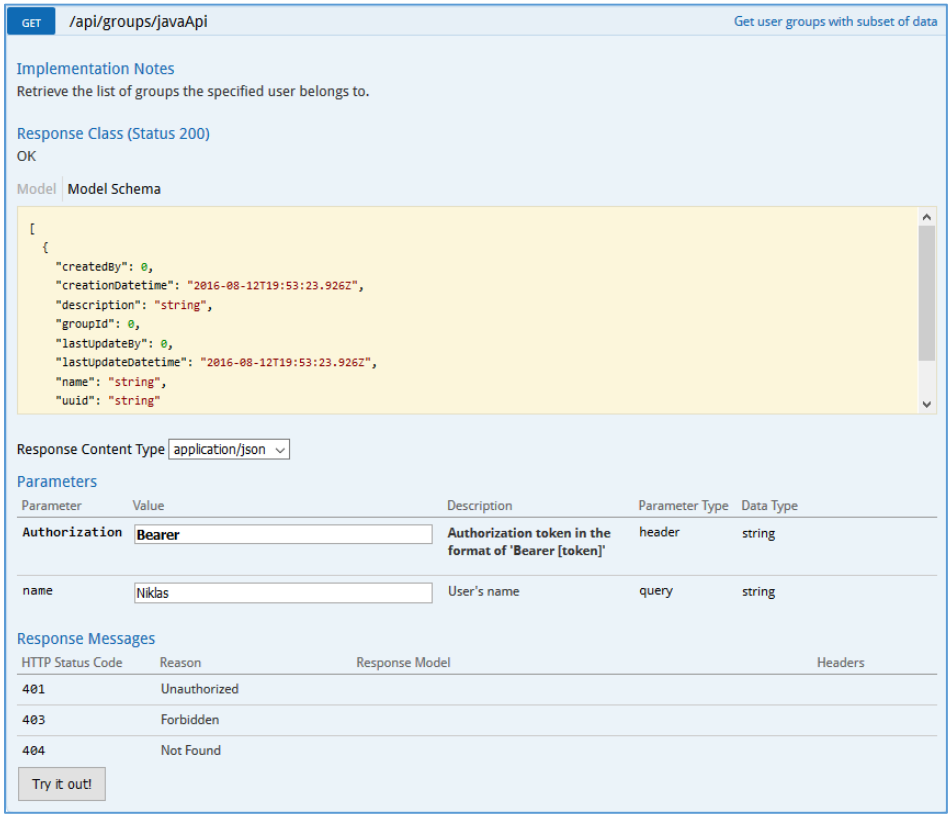
Parameter	Value	Description	Parameter Type	Data Type
Authorization	Bearer	Authorization token in the format of 'Bearer [token]'	header	string
codeSetId	10000	Code set id	path	string

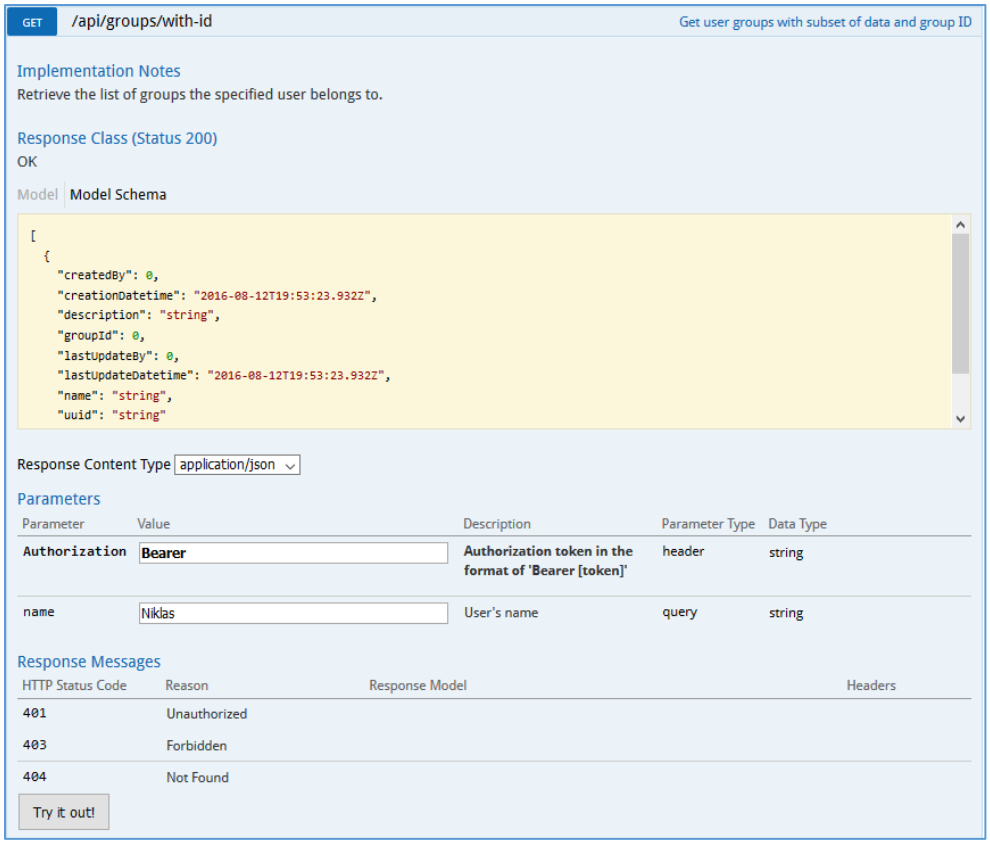
Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

5.1.2 Group and Users







GET /api/groups/with-id — Get user groups with subset of data and group ID

Implementation Notes
Retrieve the list of groups the specified user belongs to.

Response Class (Status 200)
OK

Model | Model Schema


```
[
  {
    "createdBy": 0,
    "creationDatetime": "2016-08-12T19:53:23.932Z",
    "description": "string",
    "groupId": 0,
    "lastUpdateBy": 0,
    "lastUpdateDatetime": "2016-08-12T19:53:23.932Z",
    "name": "string",
    "uuid": "string"
```


Response Content Type: application/json

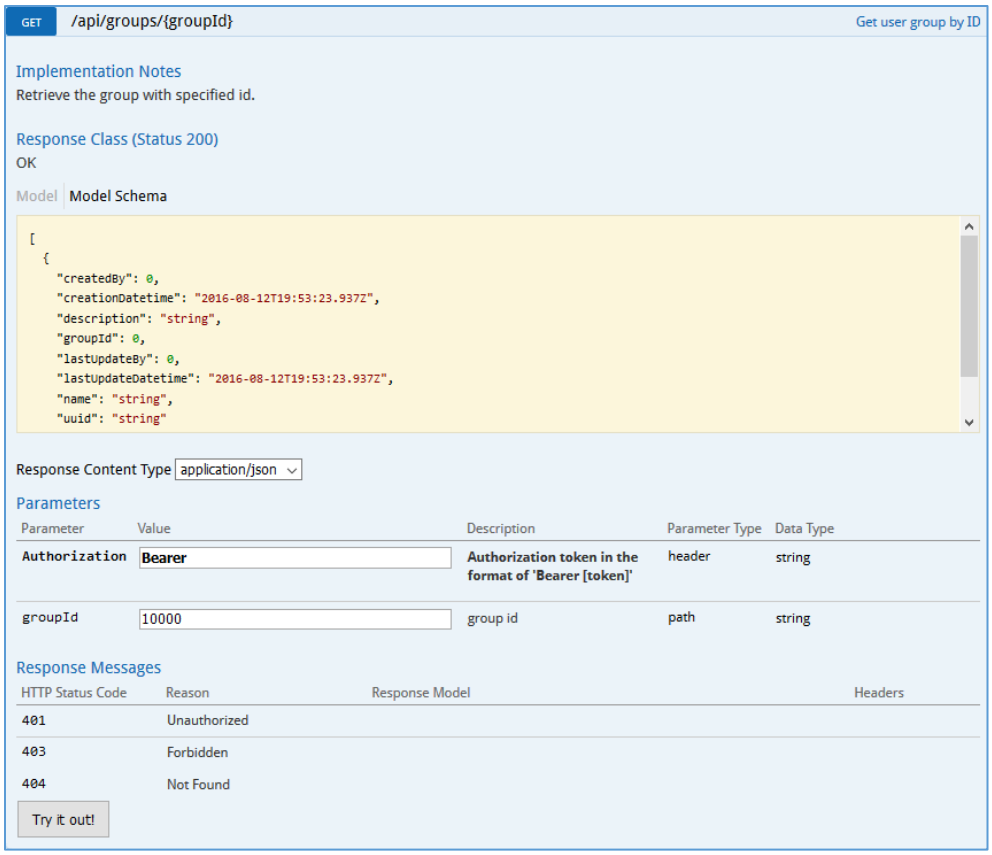
Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	**Bearer**	**Authorization token in the format of 'Bearer [token]'**	header	string
name	Niklas	User's name	query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!



GET /api/groups/{groupId} — Get user group by ID

Implementation Notes
Retrieve the group with specified id.

Response Class (Status 200)
OK

Model | Model Schema


```
[
  {
    "createdBy": 0,
    "creationDatetime": "2016-08-12T19:53:23.937Z",
    "description": "string",
    "groupId": 0,
    "lastUpdateBy": 0,
    "lastUpdateDatetime": "2016-08-12T19:53:23.937Z",
    "name": "string",
    "uuid": "string"
```


Response Content Type: application/json

Parameters

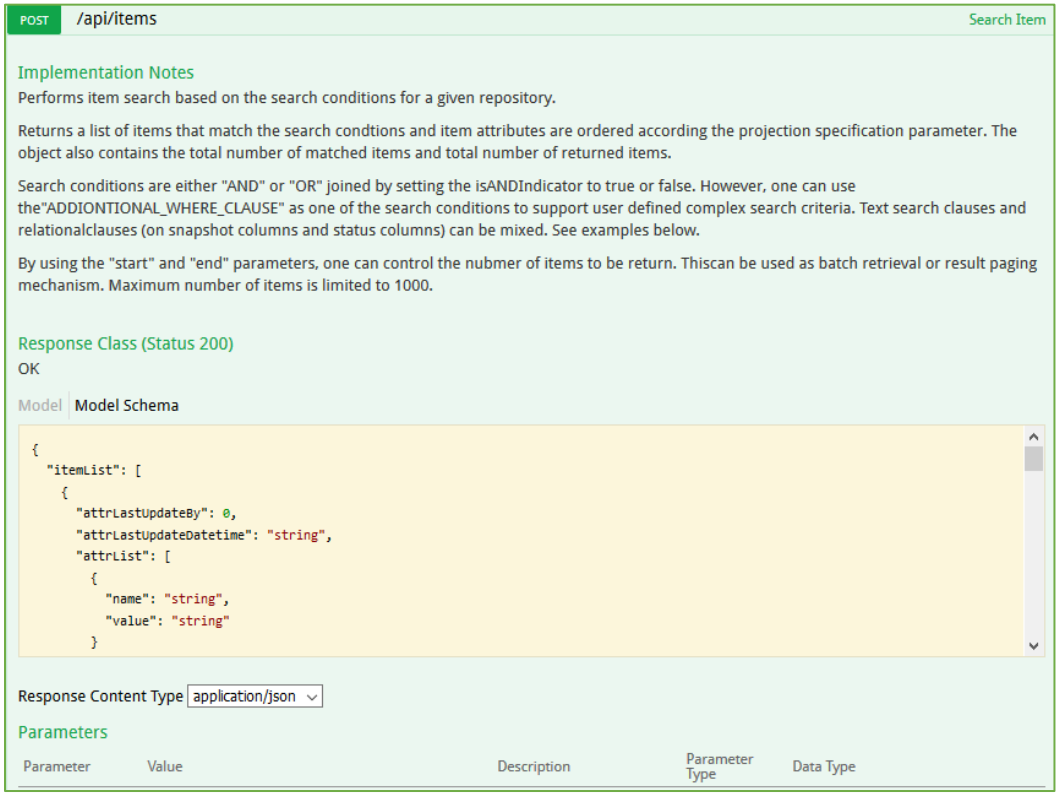
Parameter	Value	Description	Parameter Type	Data Type
Authorization	**Bearer**	**Authorization token in the format of 'Bearer [token]'**	header	string
groupId	10000	group id	path	string

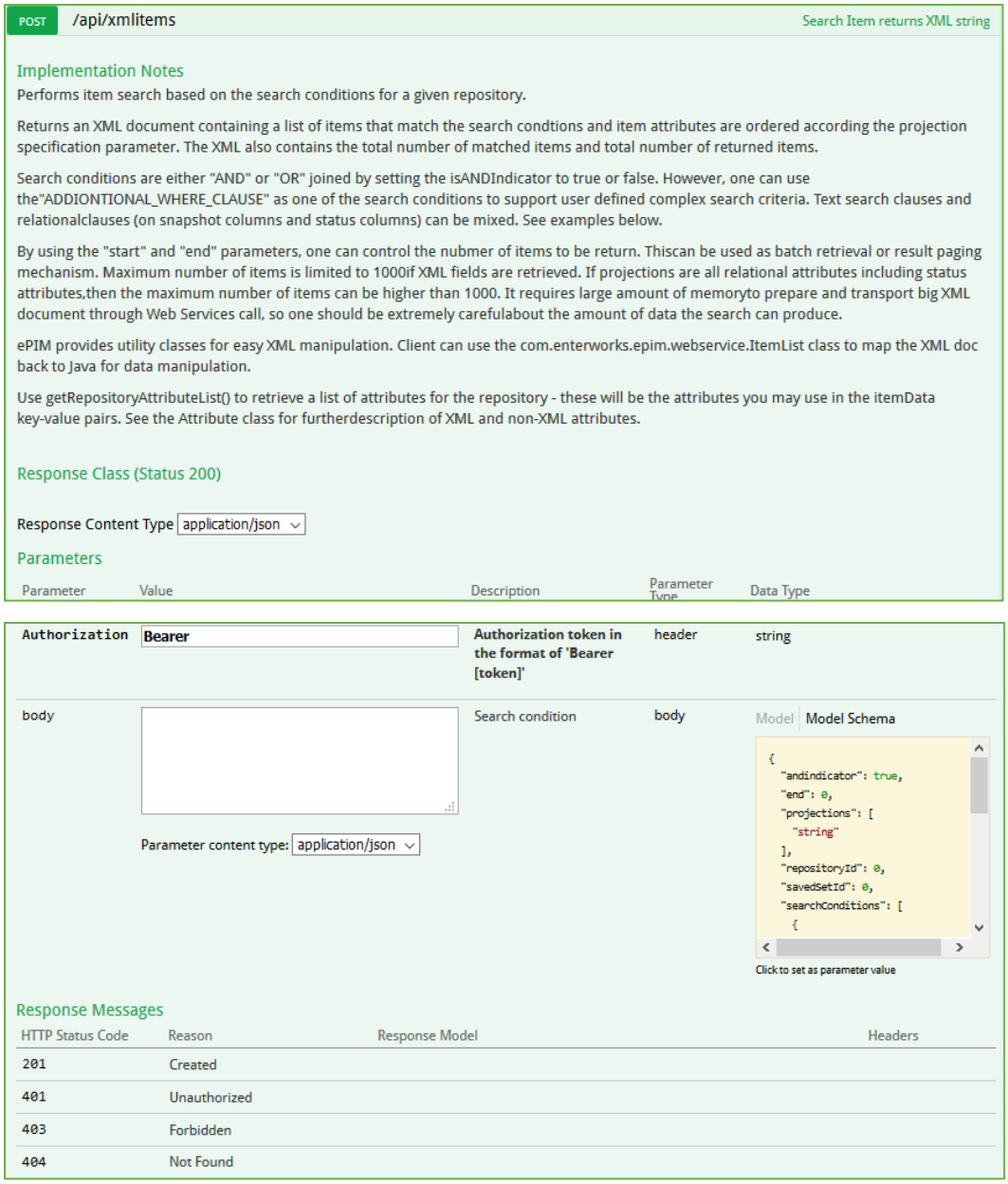
Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

5.1.3 Items


POST
/api/items
Search Item
Implementation Notes
Performs item search based on the search conditions for a given repository.
Returns a list of items that match the search condtions and item attributes are ordered according the projection specification parameter. The object also contains the total number of matched items and total number of returned items.
Search conditions are either "AND" or "OR" joined by setting the isANDIndicator to true or false. However, one can use the"ADDIONTIONAL_WHERE_CLAUSE" as one of the search conditions to support user defined complex search criteria. Text search clauses and relationalclauses (on snapshot columns and status columns) can be mixed. See examples below.
By using the "start" and "end" parameters, one can control the nubmer of items to be return. Thiscan be used as batch retrieval or result paging mechanism. Maximum number of items is limited to 1000.
Response Class (Status 200)
OK
Model
Model Schema
{
"itemList": [
{
"attrLastUpdateBy": 0,
"attrLastUpdateDatetime": "string",
"attrList": [
{
"name": "string",
"value": "string"
}
Response Content Type
application/json
Parameters
Parameter
Value
Description
Parameter Type
Data Type
Authorization
Bearer
Authorization token in the format of 'Bearer [token]'
header
string
body
{
"andindicator": true,
"end": 0,
"projections": [
"string"
],
Search condition
body
Model
Model Schema
{
"andindicator": true,
"end": 0,
"projections": [
"string"
],
"repositoryId": 0,
"savedSetId": 0,
"searchConditions": [
{
Parameter content type:
application/json
Click to set as parameter value
Response Messages
HTTP Status Code
Reason
Response Model
Headers
201
Created
401
Unauthorized
403
Forbidden
404
Not Found



POST /api/xmlitems — Search Item returns XML string

Implementation Notes

Performs item search based on the search conditions for a given repository.

Returns an XML document containing a list of items that match the search condtions and item attributes are ordered according the projection specification parameter. The XML also contains the total number of matched items and total number of returned items.

Search conditions are either "AND" or "OR" joined by setting the isANDIndicator to true or false. However, one can use the"ADDIONTIONAL_WHERE_CLAUSE" as one of the search conditions to support user defined complex search criteria. Text search clauses and relationalclauses (on snapshot columns and status columns) can be mixed. See examples below.

By using the "start" and "end" parameters, one can control the nubmer of items to be return. Thiscan be used as batch retrieval or result paging mechanism. Maximum number of items is limited to 1000if XML fields are retrieved. If projections are all relational attributes including status attributes,then the maximum number of items can be higher than 1000. It requires large amount of memoryto prepare and transport big XML document through Web Services call, so one should be extremely carefulabout the amount of data the search can produce.

ePIM provides utility classes for easy XML manipulation. Client can use the com.enterworks.epim.webservice.ItemList class to map the XML doc back to Java for data manipulation.

Use getRepositoryAttributeList() to retrieve a list of attributes for the repository - these will be the attributes you may use in the itemData key-value pairs. See the Attribute class for furtherdescription of XML and non-XML attributes.

Response Class (Status 200)

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	Bearer	**Authorization token in the format of 'Bearer [token]'**	header	string
body		Search condition	body	Model / Model Schema

Parameter content type: application/json

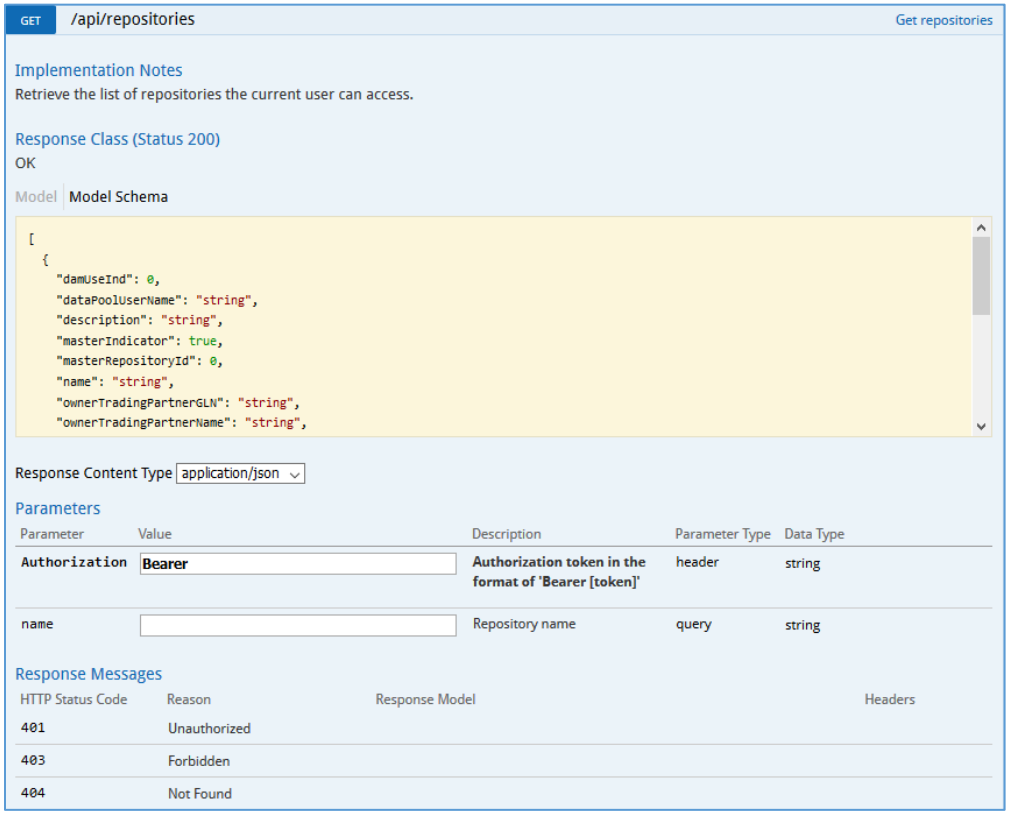

```
{
  "andindicator": true,
  "end": 0,
  "projections": [
    "string"
  ],
  "repositoryId": 0,
  "savedSetId": 0,
  "searchConditions": [
    {
```

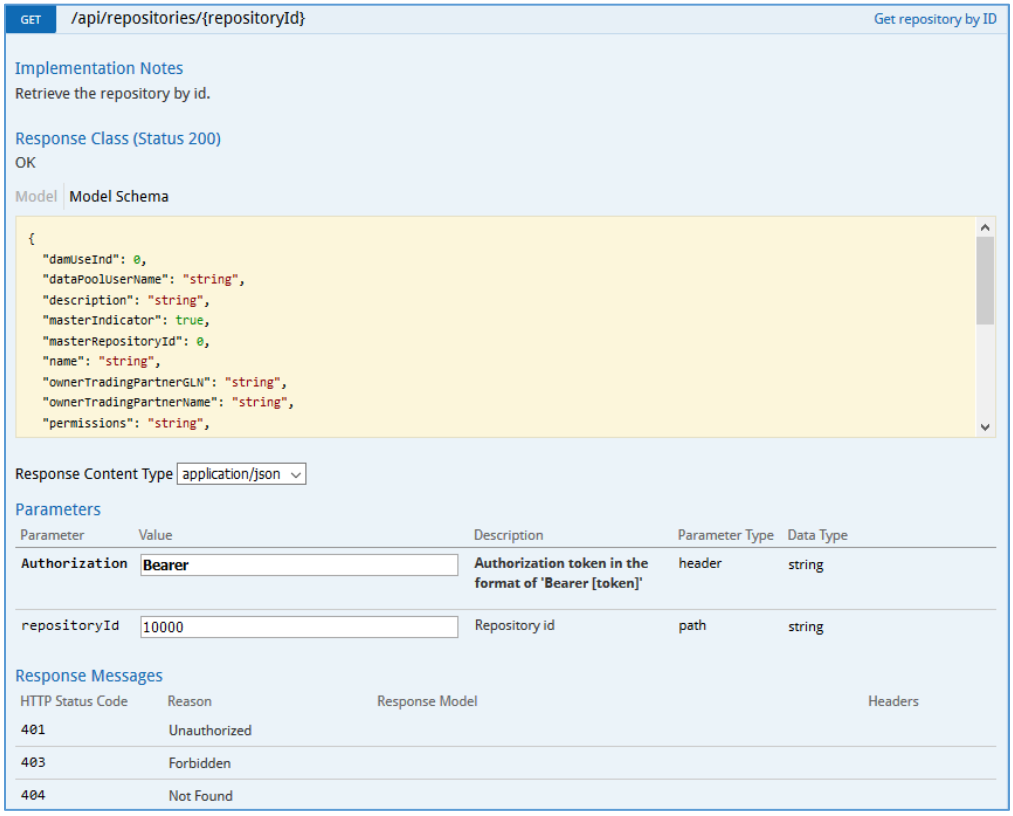

Click to set as parameter value

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

5.1.4 Repositories





5.1.5 Services

POST /api/login — login

Response Class (Status 200)
OK

Model | Model Schema

```
{
  "externalSessionInfo": "string",
  "language": "string",
  "sessionId": 0,
  "userId": 0
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	**Bearer**	**Authorization token in the format of 'Bearer [token]'**	header	string
name	(required)	**name**	query	string
pwd	(required)	**pwd**	query	string
allowexpire	(required)	**allowexpire**	query	string
language		language	query	string
externalSessionInfo		externalSessionInfo	query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

GET /api/logout — logout

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	**Bearer**	**Authorization token in the format of 'Bearer [token]'**	header	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	OK		
401	Unauthorized		
403	Forbidden		
404	Not Found		

5.2 Calling the Rest Services

The ePIM RESTful Web Service can be called in 2 different ways:

URL: http://localhost:8090/webcm/rest/groups

It will return JSON objects for all groups in the format of:

```
[{     "name": "01_Peer_Training",
    "description": ""   },
  {     "name": "22deb",
    "description": "e1"
  }]
```

Java API: public GroupDTO[] getGroupList(TokenDTO token) throws RemoteException

It will return a list of GroupDTO objects containing the same data as when calling from URL.

Base url is : http://localhost:8090/webcm/rest

The following are sample endpoints with details for review:

http://localhost:8090/webcm/rest/groups - to retrieve all groups, group by name, or group by group_id

http://localhost:8090/webcm/rest/codesets - retrieve all codesets, codeset by name, or codeset by code_set_id

http://localhost:8090/webcm/rest/repositories - retrieve all repositories, repository by name, or repository by repository_id

http://localhost:8090/webcm/rest/items – to search for items using search conditions

5.3 Rest Service Sample Code

These endpoints can be easily accessed via java with the following:

```
SessionDTO session = null;
token = restFacade.login("epim-client", "epim-client", userName, password,
null, null);

CodeSetDTO[] objList = restFacade.getCodeSetList(token);
CodeSetDTO codeSet = restFacade.getCodeSet(token, id);
GroupDTO[] objList = restFacade.getGroupList(token);
GroupDTO obj = restFacade.getGroupById(token, id);
RepositoryDTO[] repoList = restFacade.gerRepoList(token);
RepositoryDTO obj = restFacade.getRepositoryByName(token, name);
String xmlItem = restFacade.searchItemBySavedSet(token, repo, savedSet,
searchConditions, sortConditions, isANDIndicator, projections, start, end)
```

6 EJB Services

The following sections describe what are included in the Enable EJB API and how to use them.

The Enable EJB API provides an extension to allow third party applications, not just limited to Java based applications, to integrate with the Enable system. It provides most of the run-time support for the PIM management functionality. However, it is not an alternative to the web browser based user interface because it lacks most of the administrative capabilities such as user, security, repository mapping management etc.

The Enterworks Product Information Manager architecture is based on the Java 2 Enterprise Edition (J2EE) platform, and as such makes use of several J2EE platform technologies such as Enterprise Java Beans (EJBs), Java Servlets, and Java Messaging Service (JMS). Custom applications developed for Enable must be written in Java, but do not necessarily require knowledge of the internal usage of J2EE technologies employed by Enable. The Enable public API available to custom applications are written entirely in Java and do not require JMS programming skills or knowledge of EJBs or Java Servlets. These Java APIs are actually libraries of compiled codes that can be used in Enable.

6.1 Interface

The interface provides the venue by which independent systems communicate with each other. The Enable interface is EJB programming language independent and Operating System independent.

The Enable EJB APIs are described in the pim.wsdl (Web Service Description Language). The Enable installer already bundles the necessary java class files during the installation; thus, there is no need to re-generate the java client from the WSDL file. However, this is quite acceptable to do so. See the "How to Use" section for details on how to use the pre-bundled java classes. Non-Java applications can use the WSDL file as the basis to generate their corresponding client access commands in their native languages.

The core interface in the Enable EJB API is called "Server". Others are simply supporting data types that are used to either support the SOAP protocol or used as information holders.

The supporting data types are as follows:

- AttrCodeSetMapping
- AttrData
- Attribute
- CodeSet
- CodeSetDetail
- EJBServer
- EnableTreeNode
- Group

- Item
- ItemHistory
- ItemHistoryDiff
- ItemList
- ItemState
- KeyValuePair
- Media
- MediaGroup
- MediaReference
- Package-frame
- Package-summary
- Package-tree
- Repository
- RepositoryGroup
- RepositoryViewMapping
- SearchCondition
- Server
- ServerSoapBindingImpl
- Session
- Subscription
- TextSearchConfig
- TextSearchConfigAttr
- TradingPartner
- TransactionStatus

6.2 UserPreferenceServices

The Enable 7 EJB API are divided into three major functional groups: metadata retrieval, repository/item management and tools.

6.2.1 Metadata retrieval

- getCodeSet - Retrieves the metadata information for the code set.
- getCodeSetList - Retrieves a list of code sets the user can access.
- getCodeSetDetail - Retrieves the code set detail including code, code description, parent code, and other information.
- getLowerLevelCodeSetDetail - Retrieves the lower level code set detail for a particular code set and a code.
- getCustomerList - Retrieves a list of customers. The return list construction depends on the user's access rights.

- getCustomerListBySupplierGLN - Retrieves a list of customers for a particular supplier.
- getSupplierList - Retrieves a list of suppliers. The return list construction depends on the user's access rights.
- getGroupList - Retrieves the list of groups the current user belongs to.
- getRepositoryList - Retrieves the list of repositories the current user can access.
- getRepositoryAttributeList – Retrieves the attribute list for a particular repository.
- getUserPreferenceList - Retrieves the user preferences the current user can access.
- getUserPreferenceAttributeList - Retrieve attributes for a particular user preference.
- getJobStatus - Retrieves the job status.
- getAttrCodeSetMapping – Retrieves the attribute and code set mapping for a particular repository.
- getCustomerList - Retrieves a list of customers.
- getCustomerListBySupplierGLN - Retrieves a list of customers for a particular supplier.
- getEnableTree - Retrieves enAble tree nodes.
- getMediaContent - Retrieves media content for a particular media object identified by media ID.
- getMediaGroupList – Retrieves Media Groups.
- getMediaReferenceList - Retrieves references for a particular media object identified by media ID.
- getRepositoryGroupList - Retrieves the repository group list.
- getRepositoryListByGroupName - Retrieves a list of repository objects for a particular repository group.
- getSupplierList - Retrieves a list of suppliers.
- getViewMappingList - Retrieves an array of RepositoryViewMapping objects for a repository.
- UpdateMediaMetaData – Updates the media metadata only and not the content.

6.2.2 Repository/Item Management

- getItem - Retrieves the item detail.
- getItemState - Retrieves the item state.
- createItem - Creates a new item for a particular master repository.
- delistItem - Mark item delist. DeleteItem - Deletes a particular item.
- updateItem - Updates the item data

- updateItems - Updates items with search conditions.
- searchItemByRepository - You can search items in a particular repository based on simple search conditions such as "AND" or "OR" joined.
- searchItemByUserPreference - You can search items for a particular user preference.
- setItemState - Updates the item state.
- setDataPoolResponseStatus – Updates the item status based on data pool response message identifier.
- setDataPoolReponseStatusByFile – Updates the item status based on data pool response message file name.
- setDataPoolReponseStatusByTransaction - Updates the item status based on data pool response message identifier and transactions.
- validateRepository - Performs validation for a particular repository.
- syncOutRepository - Performs the sync out operation for a particular repository.
- createCodeSetDetail - Creates a code set detail record for a particular code set.
- DeleteCodeSetDetail - Delete a code detail record for a particular code set.
- isValidParent - Will interrogate the hasErrorInd of the parent item and all of its linked children with the given link relationship and return the results.
- UpdateCodeSetDetail - Updates a code set detail record for a particular code set.
- UpdateMediaContent - Updates media content only without the associated meta data.
- UpdateMediaContentByURL - Updates media content only, without the associated metadata, using content URL.
- GetParentRepositoryLinkList - Retrieves an array of RepositoryLink objects for a parent repository.
- GetChildRepositoryLinkList - Retrieves an array of RepositoryLink objects for a child repository.

6.2.3 Tools

- Login - Logs into e-PIM server to obtain a valid session.
- Logout - Logs out from e-PIM server and terminates user session.
- ChangeUserPassword - Changes the password for the login user.
- loginByGID - Log into Enable 7 server using global ID.

The method by which EJB can invoke these functions is provided in the Enable integrated Javadoc, which provides the necessary documentation for Enable 7 Web Services. The EJB

Javadoc is the generated zip file containing the API documentation of Enable's automated resources accessed via the Internet. It is extracted from the comments in the Java source code and made available in HTML format.

For more information on Java methods and calls, particularly metadata retrieval, repository/item management and tools, refer to the Enable 7 Javadoc. To access the Javadoc, navigate to the directory below and locate the wsdoc file:

```
C:\Enterworks\ENABLE\documents
```

6.3 EJB Sample Code

Java source codes are compiled into Java Class files that can be loaded to execute Methods needed to perform the different functions and capabilities of the Enable 7 application, particularly the EJB component. Presented below is the sample code or script that executes some Enable 7 methods described above.

To obtain session, the sample code snippet below can be used:

```
com.enterworks.Enable.test.webservice.Server server = new
com.enterworks.Enable.test.webservice.ServerSoapBindingImpl();
com.enterworks.Enable.test.webservice.Session session = server.login("user",
"password", true, null);

/* Demonstrate sync in, sync out and validation method calls using Enable Web
services API * /

public class PimEJBClientSample {

 private static String usage = "Usage: java PimEJBClient syncout|validataion
<repository name>";
 private static Server server = new ServerSoapBindingImpl();
 public static void main(String[] args) {

  if (args.length < 2) {
   System.err.println(usage);
   return;
  }
  if (args[0].equalsIgnoreCase("syncout")) {
   System.out.println("Sync out repository:" + args[1]);
  } else if (args[0].equalsIgnoreCase("validation")) {
   System.out.println("Validation repository:" + args[1]);
  } else {
   System.err.println(usage);
   return;
  }
  try {

   // login first

   com.enterworks.Enable.test.webservice.Session mySession =
```

```
    server.login("user", "password", true, null);

    // validate result

    System.err.println("login session:" + mySession.getSessionId());
    System.err.println("login user:" + mySession.getUserId());

    // retrieve repository list

    Repository[] repList = null;

    Repository rep = null;

    repList = server.getRepositoryList(mySession);

    if (repList == null) {
     System.out.println("No repository to retrieve.");
     return;
    }
    for (int i = 0; i < repList.length; i++) {
     if (args[1].equalsIgnoreCase(repList[i].getName())) {
      System.out.println("Find matching repository:" + args[1]);
      rep = repList[i];
      break;
     }
    }
    if (rep == null) {
     System.err.println("No matching repository found.");
     return;
    }
    if (args[0].equalsIgnoreCase("syncout")) {
     server.syncOutRepository(mySession, rep, false);
    } else if (args[0].equalsIgnoreCase("validation")) {
     server.validateRepository(mySession, rep);
    } else {
     System.err.println(usage);
     return;
    }

    //logout

    server.logout(mySession);
   } catch (Exception e) {
    System.err.println(e.getMessage());
   }
  }
```