Rival Onboard Web Service Guide

2020.2.1

11/07/2023

Rival

311 W Monroe St., Suite #402

U.S.A.

1 (866) 329.3363 U.S. toll free

Contents

Overview

Rival Onboard provides a web service for administrators to use Onboard features through an API. Onboarding methods exist in these general categories of features:

- An interface to retrieve, import, and edit user information
- An interface to launch events and retrieve event and task data
- An interface for retrieve eForm data, eForm PDFs, and uploaded documents
- An interface for category uploads

Note: This single document combines three guides that were previously published: Web Services Guide (RCWebService), Bulk User Upload and Edit (RCBulkUserImportandEdit), and Category Import (RCCategoryImport).

Methods to retrieve employee data

Four methods are available to retrieve employee data. One method returns a list of Onboarding user IDs, and the other three accept a user ID and return data about the specified user.

- The Onboarding method to return a list of user IDs is available to retrieve all users in the system or only users modified since the last retrieval.
- Three Onboarding methods are available to retrieve Onboarding employee information based on user ID. All three methods provide detailed output including employee profile data, extended user profile (custom) data, key property attributes, assignment category attributes, team membership, and event information. One method outputs string arrays, and the other two methods output an XML record.

Methods to import and update employee data

RedCarpet allows new users to be created and existing users to be edited through a user import. An import method is provided to
create new users (with or without a lifecycle event) and an edit method is provided for existing users.
The RedCarpet user import feature can be used for all employees. New employees can be created as pending users with very minimal information. Pending users are then reviewed, and approved as active users through the RedCarpet user interface. Alternatively, if all required data is available at the time of the import, new employees can be created, and established immediately as active users without a review.
Existing active employee information can be edited via the API which (also) passes an XML record to update employee information.

Methods to launch events, retrieve forms, and retrieve documents

The Onboarding API includes methods to launch and update a LifeCycle event and retrieve RedCarpet forms.

LifeCycle Event API

LifeCycle event methods are available to launch new lifecycle events or update existing events. As noted above, the user import API allows for an event to be launched when an employee is created. The launch and update event methods accommodate event management on existing employees. Methods to retrieve tasks, add tasks, add task notes, and delete tasks from existing events are also available.

Forms API

An Onboarding implementation includes the Eprise web service. It allows access to retrieve and delete saved forms outside of the Onboarding application. Available web methods include options to:

- Retrieve completed forms based on various filter criteria
- Retrieve PDF files (populated through Onboarding)
- Indicate whether a form has already been retrieved
- Delete existing forms

Documents API

An Onboarding implementation includes the Eprise web service. It allows access to retrieve and delete uploaded documents outside of the Onboarding application. The available web methods include options to:

- Retrieve uploaded documents based on various filter criteria
- Delete uploaded documents

Employees can be imported to benefit from an event and complete tasks on behalf of another employee.

Upload category data

Onboarding category upload methods allows administrators to import and update category values and modify the category structure hierarchy. It makes the maintenance of category values easier.

Administrators can create and update categories and category values through the Manage Categories menu option in the UI. They can also import category values using XML syntax via Import Category Values menu option.

Administrators can update and replace category values. The replace function lets you move child nodes and restructure the category value tree.

Getting started

Using Onboarding methods

Onboarding methods (background)

Onboarding uses Eprise as an underlying content management engine. Eprise stores user date, handles form templates and saved forms, and builds/manages the integrated Onboarding portals. The Eprise database handles all of Onboarding content maintenance and security. This is transparent to users. As an administrator using the Onboarding API, you use the Eprise web service to consume methods applicable to Onboarding.

Web Service URL

The web service is referenced with the following URL:

```
https://<hostheader>/eprise/WebServices
```

ⓘ The API references the naming convention of the underlying content management engine called "Eprise".

A description of the available methods of the Eprise web service is:

```
https://<hostheader>/eprise/WebServices?WSDL
```

While many of these methods do not apply to Onboarding, you can use this page for verification of applicable methods and their parameters.

ⓘ Note: All RedCarpet pages and API requests are served over SSL/TLS.

Logging in to consume the service

While the WSDL is public, consuming Onboarding services requires authentication.

The first parameter of each available method is a valid Eprise session ID. A valid Eprise session ID is obtained by logging in as an Onboarding employee with the correct privileges. All Onboarding methods are constrained by the same security (authentication and authorization) rules as the Onboarding user interface (performing the same function).

Best practice: Log in. Execute the applicable methods. Log out.

ⓘ Note: Onboarding role membership (and privilege assignment) is set up by your Onboarding systems administrator.

Login

Usage

Login to obtain a valid session ID.

Parameters

Parameter	Required	Description	Type
strLogInId	☐	Valid Login ID of a privileged Onboarding user account. The required employee privileges correlate to the privileges required for each method executed.	string
strPassword	☐	Corresponding password	string
strRemoteIP		In most cases this parameter does not apply to Onboarding methods. Populate this parameter only if the user is authenticated by an external authentication server. If applicable, the strRemoteIP expects ONE machine IP Address of authentication server ("remote browser").	string

Returns

- **sessionID** (simple string) — valid Eprise session id
- **""** (empty string) — unsuccessful login

Example

```
PSWebService service = new PSWebService();
 service.Url = "https://example.silkroadtech.com/eprise/ WebServices";
 String sessionnum = service.LogIn("Jim.Smith","$xxx123$","");

 if (sessionnum == "")
 {
 Console.WriteLine("Could not log in. Be sure password and URL are correct");
 return;
 }
```

Logout

Usage

Log out to terminate the session.

Parameters

Parameter	Required	Description	Type
strSessionNum	☐	Valid SessionNum returned by the Login ID method. Should be called for each cooresponding Login call.	string

Returns

- <int>1</int> — successful logout

Example

```
1  S = service.LogOut(sessionnum)
```

XML schema files

XSDs are available for all eForms and methods added to the product since 2014.

XSDs for eForms

Customers pulling Onboarding eForm data to integrate with other systems use GetFormXML to pull form data from completed forms. All eForm XSDs are available through the Onboarding user interface.

Onboarding administrators, who are members of any team with the Manage eForm List Values privilege, have access to a Download XSD button for each eForm used by your site. Your site's custom employee profile XSD is also available for download. The custom employee profile XSD defines the custom fields in GetProfileEx2 output and the custom fields available in BulkUserUpload and XMLUserEdit methods.

Log into Onboarding to access eForms and extended profile XSDs.

Methods with XSDs

XSDs to use to work with Onboarding methods are available from your site via URI. Each XSD available as:

```
https://<your sitename>/rcng/RCHelp/XSDs/<methodname and type>.xsd
```

For example if your site url is **abc-redcarpet.silkroad.com**, an XSD can be located at:

```
https://abc-redcarpet.silkroad.com/rcng/RCHelp/XSDs/GetTasksInput.xsd
```

The following list of XSD are available:

If you access the URL https://<your sitename>/rcng/RCHelp/XSDs/, without an XSD from the list below, you will be presented with the available XSDs.

Method Name	XSD
AddTaskEx2	AddTaskEx2Input.xsd
	AddTaskEx2Results.xsd
AddTaskNoteEx	AddTaskNoteExInput.xsd
AssignTask	AssignTaskInput.xsd

Method Name	XSD
CompleteTaskEx	CompleteTaskExInput.xsd
GetCompletedFormsList	GetCompletedFormsListInput.xsd
	GetCompletedFormsListOutput.xsd
GetEventReportEx	GetEventReportExInput.xsd
GetTasks	GetTasksInput.xsd
	GetTasksOutput.xsd
GetUploadedDocumentList	GetUploadedDocumentListInput.xsd
	GetUploadedDocumentListOutput.xsd
GetUserIDList	GetUserIDListInput.xsd
	GetUserIDListOutput.xsd
GetUserProfileEx2	GetUserProfileEx2Input.xsd
ReopenTask	ReopenTaskInput.xsd

Employee ID Types

Users are uniquely identified in Onboarding based on your implementation.

You can query user data based on the Onboarding user for your implementation. All of these options are returned by GetUserIdList. The following criteria can be used to identify a user:

Note: The terms *user* and *employee* mean the same thing.

ID	Required	Unique	Editable with:	Lookup Parameter for:	Notes
LoginID	✓	✓	BulkUserUpload XMLUserEdit	GetUserProfile GetUserProfileEx GetUserProfileEx2 XMLUserEdit GetUploadedDocumentList	All users in Onboarding must have a LoginID. This is the primary unique identifier throughout the system.
Email	⚠		BulkUserUpload XMLUserEdit	GetUserProfile GetUserProfileEx GetUserProfileEx2 XMLUserEdit GetUploadedDocumentList	Email address can be configured to be required - **Default** is required. Email address can only be used to lookup users if it is unique to one user account. When using SilkRoad Authentication the email address is the user name, therefore is required and must be unique.
Guid		✓		GetUserProfile GetUserProfileEx GetUserProfileEx2 XMLUserEdit GetUploadedDocumentList	A (Onboarding) generated system key. Guid is unique for Onboarding users. It is not exposed through the Onboarding user interface. Note: For GetUserProfile and GetUserProfileEx, GUID can be used in the strLoginID field.

ID	Required	Unique	Editable with:	Lookup Parameter for:	Notes
LifeSuiteID		✅		GetUserProfileEx2 GetUploadedDocumentList	A (SilkRoad Technology) generated system key. It is not exposed through the Onboarding user interface. This key exists for integrating user data across SilkRoad applications. **Not** included in the output of GetUserProfile or GetUserProfileEx
Employee_HRISID	⚠️	✅	BulkUserUpload XMLUserEdit	GetUserProfileEx2 XMLUserEdit GetUploadedDocumentList	Employee_HRISID can be configured to be required - **Default** is *not* required. The user interface label can be configured when used it must be unique. This field exists to cross reference an employee with their ID in another (corporate) system. SilkRoad recommends you populate this value for system cross reference. **Not** included in the output of GetUserProfile or GetUserProfileEx

ID	Required	Unique	Editable with:	Lookup Parameter for:	Notes
SSOAuthParam	⚠	✓	BulkUserUpload XMLUserEdit	GetUserProfileEx2 XMLUserEdit GetUploadedDocumentList	Required when a user account is configured to use external authentication such as Single Sign-On (SSO). The user interface label can be configured. The authentication parameter is the user ID used to authenticate in an external system. Customers can choose to populate the SSOAuthParam with the external system identifier even if they are not using external authentication. Synonymous with "AuthParam" for BulkUserUpload and XMLUserEdit

The unique identifiers listed above appear in the "Details" section of the employee record.

Localization

Localized objects in SilkRoad Onboarding include the following labels:

- Abstract category for task assignment categories
- Category alias for key properties
- Named person for key properties and events
- Benefiter title for event definitions
- Dates for key properties and events

Each of these objects can be configured for a localized display at the administrator level. Multi-lingual (string) values can be viewed and maintained on the Localization page in the Onboarding interface. Pages in Onboarding are rendered based on a viewer's browser language setting. System level strings are localized as part of the base configuration, and administrator level strings are defined on Localization page.

Each object has a unique code. The unique code (viewed on the Localization page) allows the object to be identified programmatically. The <Code> node is synonymous to the <Name> node in the XML syntax. The code field is unique and independent of language. The <Name> value is localized.

The code option provides unique identifier for categories on bulk import, XML user edit, launch event, and update event, and the portal event filter to identify localized objects. Before availability of a code, the object was identified by name and was specified in the default language.

Note the use of the <Code> node in lieu of the <Name> node applies to programmatically setting a value in a particular scenario. Examples of scenarios are programmatically adding or deleting assignment categories to an employee, adding or deleting the key property categories for an employee, adding or deleting the key property people or dates for an employee, adding an event for an employee specifying named people, benefiter title, and dates.

Onboarding generates a unique code for each object, and the codes can be updated on the Localization page. The codes are included in the XML export on the Import/Export Category Values page.

On upgrade, if a code identifier is not present, a default code identifier is generated. You can view all codes on the Localization page.
The XML syntax for following methods:

- BulkUserUpload
- XmlUserEdit
- LaunchEvent
- UpdateEvent

allow the substitution of a <Code> </Code> node for the occurrence of the <Name> </Name> node. The <Code/></Code> node should be used while specifying an abstract category, category alias, named person, event, or date.

> Administration>Localization menu option is available to team members of teams with the Manage Localization privilege.

You can also specify the code value instead of the name value when calling the GetFormsIDsEx method. It is acceptable to pass the code node in the strEventName and the arrCategoryFilter parameters.

Example syntax

There are resources in the Onboarding Administration interface to confirm the code values Onboarding is expecting. One is the Localization page, which allows you to update the code values (to match your external system). Another is the Import/Export Category page, which allows you to export all the defined categories with their corresponding values.

You can reference your Locations page for the type codes ("category alias") in this example. If you are updating a category the corresponding codes can be verified through the import/export for the (underlying) category values. In this example "Location" is assigned a value for a new employee. The code for the category alias "Location" is noted <Code>Category_1</Code. A snapshot of a (English and French) localization page is:

	Type	Code	Default language	Other Languages
	Abstract Category	LocationsCategory_Code	Locations	Français, русский язык, 中文 (简体), Português
	Abstract Category	DepartmentsCategory_Code	Departments	Français, русский язык, 中文 (简体), Português
	Abstract Category	JobsCategory_Code	Jobs	Français, русский язык, 中文 (简体), Português
	For Whom Title	BenefiterTitle_4	New Hire	Français, русский язык, 中文 (简体), Português
	For Whom Title	BenefiterTitle_5	Existing Employee	Français, русский язык, 中文 (简体), Português
	Category Alias	Work_Location_Code	Work Location	Français, русский язык, 中文 (简体), Português
	Category Alias	Department_Code	Department	Français, русский язык, 中文 (简体), Português
	Category Alias	Job_Code	Job	Français, русский язык, 中文 (简体), Português
	Category Alias	Category_13	Old Location	Français, русский язык, 中文 (简体), Português
	Category Alias	Category_14	New Location	Français, русский язык, 中文 (简体), Português
	Category Alias	Category_15	Old Department	Français, русский язык, 中文 (简体), Português
	Category Alias	Category_16	New Department	Français, русский язык, 中文 (简体), Português
	Category Alias	Category_17	Old Job	Français, русский язык, 中文 (简体), Português
	Category Alias	Category_18	New Job	Français, русский язык, 中文 (简体), Português
	Date Label	Date_6	Start	Français, русский язык, 中文 (简体), Português
	Date Label	Date_7	Expiration	Français, русский язык, 中文 (简体), Português
	Date Label	Date_8	Transfer	Français, русский язык, 中文 (简体), Português
	Event Name	Event_4	Onboarding	Français русский язык, 中文 (简体), Português
	Event Name	Event_5	I-9 Reverification	Français, русский язык, 中文 (简体), Português

The syntax below is an example of using <Code> to specify the key property criteria instead of <Name>. The code corresponds to value listed for the category alias on the Localization page. Onboarding assigns unique default values ("Category_1" in this example). You can use the Edit link to update the code to something more relevant.

```
<?xml version="1.0"?>
<users>
 <user>
  <First_Name>Bonnie</First_Name>
  <Last_Name>Greggory</Last_Name>
  <LoginID>Bonnie.Greggory</LoginID>
  <Email>Bonnie.Greggory@some.com</Email>
  <ForceActiveNewHire>1</ForceActiveNewHire>
  <Password>$Bonnie.Greggory$</Password>
  <KeyProperties>
```

```
    <Category>
     <Code>Category_1</Code>
     <Value>CA</Value>
     <Type>Code</Type>
    </Category>
   </KeyProperties>
  </user>
 </users>
```

Category Name/Value/Type syntax

Methods that use this syntax

- BulkUserUpload (see page 30)
- XMLUserEdit (see page 72)
- LaunchEvent (see page 168)
- UpdateEvent (see page 174)

Referencing category values

Category values are used in a various ways in Onboarding. Name, value and type elements are used in many places.

Categories are used:

- As part of **events** to identify where an employee fits in the organization, which determines which tasks are assigned to their event.
- To identify the **key properties** of an employee, identifying properties of an employee beyond the context of an event.
- To identify an employee's **assignments** or responsibilities. (Referred to as **assignment categories**)

Name

When referencing a category value, the context of the category to be used must be established. The Name node identifies this.

Category vs Alias

When working with assignment categories, the category (or abstract category) names are used.

When working with event categories or key properties, the category aliases are used. This allows a single abstract category (collection of values) to be used for multiple purposes on an event. For example, the locations listing could be used for both work and home locations or an old and new department can be used for a transfer event using one listing of departments.

Type	Code	Default language
Abstract Category	LocationsCategory_Code	Locations
Abstract Category	DepartmentsCategory_Code	Departments
Abstract Category	JobsCategory_Code	Jobs
For Whom Title	BenefiterTitle_4	New Hire
For Whom Title	BenefiterTitle_5	Existing Employee
Category Alias	Work_Location_Code	Work Location
Category Alias	Department_Code	Department
Category Alias	Job_Code	Job

In above screen sample (from the Localization page):

- Locations, Departments, and Jobs are abstract categories that can be used when referring to assignment categories

- Work Location, Department, and Job are category aliases that can be used when referring to key properties or event categories (depending on the configuration of the specific event)

Type - Name, Code or Path

There are three methods to refer to a specific category value:

1. Name - the name of the category value
    - can only be used if the name is unique in the category
2. Code - the code that is associated with the value
    - each code can only be used once in each category
    - configuration can set code as required for all category values
3. Path - value contains path
    - the full path of names to a value.

− All Locations	All_Locations
− United States	US
− Massachusetts	MA
Boston	US_MA_BOS

In the example above, to identify "Boston" as the value the name, code or path can be used:

1. Name = "Boston"
2. Code = "US_MA_BOS"
3. Path = "All Locations\United States\Massachusetts\Boston"

Using "name" only works correctly if there are no other values named "Boston" in the location category. For this reason "Code" is the recommended identifier for category values when an integration will be used.

Element Syntax	Description
Name	Is used to identify the category Example syntax: <Name>Work Location</Name>
Value	Valid values are based on the <Type> element. Example syntax: <Value>US_MA_BOS</Value> <Type>Code</Type> is the recommended Name/Value/Type set to use.

Element Syntax	Description

Type	Valid values are:
	• "Code" - value is the category code (recommended)
	• "Name" - value is the field name
	• "Path" - value contains path
	Type = "Code"
	`<Value>US_MA_BOS</Value>`
	`<Type>Code</Type>`
	Type = "Name"
	`<Value>Boston</Value>`
	`<Type>Name</Type>`
	Type = "Path"
	`<Value>All Locations\United States\Massachusetts\Boston</Value>`
	`<Type>Path</Type>`
	<Type> determine what type of value is passed in the value field to associate the correct category.
	<Type>Name</Type> can **not** be used if duplicate values exist within your category tree.

Syntax of the Category Parameter

Methods that use this syntax

- GetFormsIDsEx (see page 133)
- GetCompletedFormsEx (see page 132)
- GetEventReport [Deprecated]

Syntax of the Category Parameter

GetFormsIDsEx and GetCompletedFormsEx allow for an optional input parameter of category values as a filter criteria for the results set. Categories defined in your Onboarding implementation are customized for your company.

The values are determined by your implementation.

Categories have multiple uses in Onboarding. In the context of forms, you can filter on event categories to retrieve specific categories of forms. The event category values selected for the launched event are associated with the forms generated for the event's tasks.

An array of array of strings are used to pass the name/value pairs for each category to be used in the filter. The array of strings contains these elements:

1. Alias (Category) name. The string should be specified as defined in the Manage Event page visible in the Onboarding user interface.
2. Unique (category) value. The string should be specified as defined in the Manage Hierarchical Category Values page in the in Onboarding user interface. The unique value can be specified in one of three formats.
    - by internal code
    - by name
    - by path in the hierarchical tree

Use one of these three values, whichever is most convenient to your configuration. The method you use must result in a unique value. If you have a job type category value such as "Staff' that exists under Exempt\Staff and Non-Exempt\Staff you may not reference "Staff" by name. You may use the path to reference the value. The path (hierarchical tree) notation uses backslashes "\\" to note the node. The format type is not specified. A unique value is the only criteria.

```
String[][] arrCategory = new String[3][];
arrCategory[0] = new String[]{"Location","Boston"};  // Specifies a name.
arrCategory[1] = new String[]{"Department", "CS_BLDNG_5"};  // Specifies a code, not a
name.
arrCategory[2] = new String[]{"Job Type", "\\Exempt\\Staff"};  // backslashes,
including leading \ to note path
```

User Import and Edit API

Overview

The user import API allows a bulk update and bulk edit of employees in the system using an XML document as the input parameter.

Onboarding provides a import feature allowing employees to be added and existing users to be edited through program interface. The import or edit is executed by passing the XML as parameter through a Web Service method.

The Onboarding program interface mimics the information collected in the Onboarding user interface. Features for creating and editing employee information available in the interface are also available through the program interface.

Pending employees and active employees

The program interface (import features) differs from the user interface as it allows partial information to be entered into the system as a pending employee" New employees can be created with very minimal information, reviewed and pending an approval, then submitted as active users through the Onboarding user interface.
Alternatively, if all required data is available at the time of the import, new employees can be created and established immediately as active users without a review.
Existing active employee information can also be updated or deleted from an XML record.

Importing new employees as pending or active

Imported users can be created in one of two ways:

- As *active* users

  New employees can be immediately created as active users. The XML record must include all the required employee data and an indicator (<ForceActiveNewHire>1</ForceActiveNewHire>) to signify the creation of an active user.

  An event can optionally be included in the record. If an event element is present in the employee record, tasks are created based on the specified category values.
  Active users can log in to Onboarding, the applicable Onboarding portals, and appear in the Find Employee List with no additional intervention.

- as *pending* users

  New employees can be created in a temporary queue if incomplete information is provided in the import record. The temporary queue is called the pending employee list. The pending employee list is a staging area allowing an approval manager to complete the employee information before activating the new user. The approval managers can use the pending employee list to approve, assign, or delete pending employees.
  See Review and approve a pending user (see page 70) for information on converting pending employees to active employees through the Onboarding user interface.

> Note: An approval manager is a user who is member of a team with Pending User privileges.

BulkUserUpload

Usage

Used to create one or more new users using an XML formatted file.

Parameters

Parameter	Required	Description	Type
strSecurityToken	✓	Valid Session ID for consuming service	simple string
xmlUserData	✓	XML record formatted as described in Chapter 1 of the User Import Guide.	simple string

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

String containing XML record with error description

Examples

To import new Onboarding users from an XML record.

```
string sUsers = "<?xml version='1.0'?><users>" +
  "<user><First_Name>Jane</First_Name><Last_Name>Brown</Last_Name>"+
  "<LoginID>Jane.Brown</LoginID>"+ "<Email>Jane.Brown05@silkroadtech.com</Email>"+
```

```
  "<ForceActiveNewHire>1</ForceActiveNewHire>"+ "<Password>$JaneBrown$</Password></user>"+
  "<user><First_Name>John</First_Name>"+ "<Last_Name>Smith</Last_Name>"+
  "<LoginID>John.Smith</LoginID>"+ "<Email>John.Smith@silkroadtech.com</Email>"+
  "<ForceActiveNewHire>1</ForceActiveNewHire>"+ "<Password>$JohnSmith$</Password></user>"+
 "</users>"

Eprise.EpsStringEx ret = service.BulkUserUpload(sessionnum,sUsers); if (ret.ErrorNum != 1)
{
 Console.Out.WriteLine("Error while trying import users" + " Error:" + ret.ErrorString);
 return;
}
```

Example return ret.Data String:

```
<?xml version="1.0"?>
<BulkImportResults>
 <Note>2 active users created</Note>
 <Note>0 pending users created</Note>
 <Note>Parsing started at Thursday, July 24, 2008 1:43:38 PM and ended at Thursday, July 24, 2008 1:43:38 PM</Note>
 <Note>Total duration: .0598 seconds (.0465 seconds were spent creating users)</Note>
 <CreatedActiveUsers>
  <CreatedActiveUser>Jane Brown</CreatedActiveUser>
  <CreatedActiveUser>John Smith</CreatedActiveUser>
 </CreatedActiveUsers>
</BulkImportResults>
```

The file import is a synchronous process with a reported return code to communicate the success of each user record. The success or failure message of an imported record can be emailed and logged.

ⓘ Note: Configuration criteria of the email for the return codes is described below

User requirements to execute BulkUserUpload

A user executing the method must:

- Be a member of a team with Pending Users privileges to create pending users
- Be a member of a team with Create User privileges to create active users
- Be a member of the event coordinator's team for the event if attempting to launch an event for a new employee

The default installation includes a Service team with these privileges assigned.

Importing a record

The user fields follow the same naming convention if they are posted from an XML record.

Element syntax

As dictated by the (standard) rules of XML syntax for XML elements:

- All elements must have a closing tag
- Opening and closing elements must be written in the same case
- XML documents must always include a root element (<users></users> in this case).
- White space is preserved
- XML tags are case sensitive

Example XML Record

```
<?xml version="1.0"?>
<users>
 <user>
  <First_Name>Jane</First_Name>
  <Last_Name>Brown</Last_Name>
  <LoginID>Jane.Brown</LoginID>
  <Email>Jane.Brown05@silkroadtech.com</Email>
  <HireDate>2011-06-01</HireDate>
  <ForceActiveNewHire>1</ForceActiveNewHire>
  <Password>$JaneBrown$</Password>
 </user>
</users>
```

The root element is <users>.

The child element, <user> occurs for each imported user.

The subchild elements reflect the fields in the user form.

This record creates an active new employee named Jane Brown with an email address of Jane.Brown05@silkroadtech.com and a login id / password Jane.Brown / $JaneBrown$

Processing results

Active users and pending users are processed differently.

If a record is creating an **active** user, all required fields on the record must contain correct values. If the record contains an unrecognized (incorrect) required value, the individual user record fails.

If the record contains an unrecognized (incorrect) optional value (such as assignment category, team membership, key properties, or shared dashboard criteria), the individual user record is created but displays a warning. By default, a warning message is generated in your output if an unrecognized node was detected. The warning messages can be suppress by including an ignorewarnings attribute on the users record.

```
<users ignorewarnings="1">
...
</users>
```

A pending user must be reviewed and approved before an active user is created. The import process for a pending user is not restricted to valid values. If a record dose not contain the ForceActiveNewHire flag, a pending user is created.

If the record contains an unrecognized node (syntax) or invalid value, a pending user is created, and incorrect tags and values are ignored.

The following is response is returned from the above example of an active user is:

BulkUserUpload Results

```
<?xml version="1.0"?>
<BulkImportResults>
 <Note>1 active users created</Note>
 <Note>0 pending users created</Note>
 <Note>Parsing started at Tuesday, May 06, 2008 12:58:27 PM and ended at Tuesday, May 06, 2008 12:58:27 PM</Note>
 <Note>Total duration: .0002 seconds (.0001 seconds were spent creating users)</Note>
 <CreatedActiveUsers>
  <CreatedActiveUser>Jane Brown</CreatedActiveUser>
 </CreatedActiveUsers>
</BulkImportResults>
```

Users with errors are included in the output with in a <BadUsers> record in the results set.

Importing a record

Any employee created as an *active* user must have all the required elements in the XML record. You must also include an indicator in the new employee user record to force the creation of an active user.

```
<ForceActiveNewHire>1</ForceActiveNewHire>
```

When attempting to create an *active* user, if required data is not included the imported record will fail.

Note: If an import record has required fields omitted, the record will fail. If a record contains an invalid value, the record will fail.

In contrast to an active user, an employee created as a pending user requires only a single element. Any element can be used to create a pending user, but specifying <First_Name>, <Last_Name>, and <Email> is highly recommended. These fields are listed in the (three) Pending User Lists and allow a permissioned Onboarding user to locate the new pending employee. The permissioned Onboarding user reviews the pending list to approve pending users. The Review and Approve process presents the user edit form where all the required attributes for an active user are populated.

Import record required fields for an active employee

Required fields apply only to employees being created as active users. The required elements for a new employee are:

```
<First_Name/>
<Last_Name/>
<LoginID/>
<ForceActiveNewHire/>
```

Note: These fields are the only required fields for creating a record. However, the configuration of the system can make additional fields required, including Email and Employee_HRISID. Custom fields can also be configured to be required.

Defining an employee

Two classes of employees are imported into Onboarding:

- New Hires - employee records with an event
- Existing Employees - employee records without an event

New employees to your organization can be imported into Onboarding and events launched for them. Many existing employees (hiring managers, HR personnel, and supporting team members) need user accounts in Onboarding to support new hires. In Onboarding the difference between employee classes is that new hires are imported with an event while existing employees are not.

When importing an employee into Onboarding, any of the follow sets of information can be included. The table below indicates the data sets that are typical (✓) for each employee type. Each implementation of Onboarding can vary. Specific requirements should be identified for each Onboarding integration.

Data set	New hire	Existing employee	Notes
Basic profile	✓	✓	
Extended profile	✓		These are custom fields for each installation. These fields are not included in this guide.
Team membership		✓	
Assignment categories		✓	
Shared Dashboard			
E-Verify information			
Key Properties	✓	✓	
Event data	✓	⛔	

The following sections go into details of importing each set of data.

After an employee is a user in Onboarding, their data can be edited using the UpdateEvent (see page 174) method for event changes or XMLUserEdit (see page 72) method any other changes. The XMLUserEdit syntax is similar to but not identical for this method.

Basic and extended profile

These fields are available in every Onboarding installation:

- First, Middle and Last name

- Email Address
- Work and Mobile phone numbers
- Login ID
- Password
- SSOAuthParam
- Employee_HRISID
- Hire and termination dates

See the Basic Employee Profile (see page 44) section for additional details on these fields.

Each Onboarding implementation includes the configuration of extended profile or custom fields. These fields collect data from new employees once then can be automatically filled in on the various forms that they are required to fill out as they come into the organization. Typically these fields include:

- Contact information such as addressee
- Personal information such as legal name or nick names
- Position information or other data from the ATS that is held for integration with an HRIS

Importing users for external authentication (SSO)

By default, users are imported as standard Onboarding users. If your implementation is configured to allow an external application (instead of standard Onboarding) to authenticate employees, you can choose to import a user for external authentication. External authentication requires <AuthType> and <SSOAuthParam> tags in the user record. Active users can be immediately created for external authentication. The following subchild elements are specified:

```
<AuthType>CUProfile</AuthType>
<AuthParam>JJones</AuthParam>
```

<AuthParam> expects the value of the unique identifier used to authenticate the user in the external application.

> Note: CUProfile means Cached User Profile. Onboarding can be configured to redirect to an external application to authenticate an Onboarding user.

The following is example syntax of creating an active user for external authentication with an onboarding event:

```
<user>
 <First_Name>James</First_Name>
 <Last_Name>Walters</Last_Name>
 <LoginID>James.Walters</LoginID>
 <Email>James.Walters@silkroadtech.com</Email>
 <HireDate>2008-08-12</HireDate>
 <ForceActiveNewHire>1</ForceActiveNewHire>
 <AuthType>CUProfile</AuthType>
 <AuthParam>James.Walters</AuthParam>
 <Event>
  <Name>Onboarding</Name>
  <Person>
   <Name>HR Coordinator</Name>
   <Value>HRCoord@silkroadtech.com</Value>
  </Person>
  <Person>
   <Name>Manager</Name>
```

```
    <Value>ManagerEast@silkroadtech.com</Value>
   </Person>
   <Date>
    <Name>Start</Name>
    <Value>2008-08-12</Value>
   </Date>
   <Category>
    <Name>Department</Name>
    <Value>Customer Service</Value>
    <Type>Name</Type>
   </Category>
   <Category>
    <Name>Job Type</Name>
    <Value>Staff</Value>
    <Type>Name</Type>
   </Category>
   <Category>
    <Name>Location</Name>
    <Value>East</Value>
    <Type>Name</Type>
   </Category>
  </Event>
 </user>
```

AuthParam as an alternate unique identifier for users

<AuthParam> expects the value of the unique identifier used to authenticate the user in the external application. If your company chooses to implement Onboarding using an external application to authenticate users, <AuthParam> is required and is used to pass to the other application for authentication.

You can also choose to populate <AuthParam> if you are using standard authentication. The <AuthParam> field is optional but can be useful for data integration purposes. If you have an external system (an HRIS application), SilkRoad recommends you populate the AuthParam with the unique identifier established in the external system. Populating the AuthParam field eases data transfer between systems regardless of the authentication process. It is available as a unique identifier for the XmlUserEdit method for Onboarding employees.

While AuthParam can be used as an alternate identifier, the Employee_HRISID field is designed specifically for this use. See Employee ID Types (see page 20) for more details on employee identifiers.

Converting a standard user to an external User

Employees imported as the default (standard) pending users can be approved to create an active user. Standard active users can be edited to be converted to external authentication at any time.

All active users are listed in the Find Employee search. By choosing **Edit Profile** from the Find Employee user list **Options** button, a permissioned user can convert the user from Standard Authentication to External Authentication. The **Authentication Parameter** is the unique identifier used to authenticate the user in the external application.

Team membership

Teams allow tasks to be assigned to groups of employees and privileges to be managed in Onboarding.

When an employee is a member of a team, they can view tasks assigned to that team and have privileges assigned to that team.

When an employee is a controller of a team, they can add and remove team members of that team but are not automatically members of that team.

If you import an active user and intend for them to automatically be assigned tasks, team membership is required. Assignment categories can be used to limit the scope of tasks assigned to an employee. You can edit their profile through Onboarding to add assignment categories or include team membership and assignment categories in the XML import record.

Team membership is noted with the parent node of <Teams> and a child node <Team> for each team the employee will be a member. By default, the <Team> node adds the employee as a member of the team. The <TeamController> node is used to specify the employee should be added as a team controller.

Example team membership XML:

```
<Teams>
  <Team>TeleSupport</Team>
  <Team>EastCoastSupervisors</Team>
  <TeamController>EastCoastSupervisors</TeamController>
</Teams>
```

Category syntax

When creating your syntax for importing employees, be aware of the different instances of category values and their meanings.

- *Event* categories apply to employees in Onboarding who benefit from tasks. Event categories are populated with a **single value** for each category. Event categories apply to the employee.
- *Assignment* categories apply to employees who exist in Onboarding as providers. Providers are users who provide some service for fellow employees. Providers are assigned tasks and/or view others tasks. The assignment categories are used in the task assignment logic. Assignment categories do not apply (are not referenced) to employees who exist in Onboarding to benefit from tasks. Assignment categories can have multiple values for each category. For example, a provider may be responsible for completing tasks on behalf of fellow employees in multiple locations.
- *Key Property* categories can apply to any employee in Onboarding. Key properties are used as a search criteria to locate employees in Onboarding. Key property values do not impact task assignment.

Users can both benefit from event(s) and be providers. In this case, the user would be created with both assignment categories and event categories.
The event categories are child noted of the <Event> element. The assignment categories are child nodes of the <Categories> element. The key property categories are child nodes of the <KeyProperties> element. Other than the parent element, the same syntax is used for category name(s) and category value(s).

The same basic syntax is used when referencing category values.

Element Syntax	Description
Name	Is used to identify the category Example syntax: <Name>Work Location</Name>
Value	Valid values are based on the <Type> element. Example syntax: <Value>US_MA_BOS</Value> <Type>Code</Type> is the recommended Name/Value/Type set to use.

Type	Valid values are:
	• "Code" - value is the category code (recommended)
	• "Name" - value is the field name
	• "Path" - value contains path
	Type = "Code"
	`<Value>US_MA_BOS</Value>`
	`<Type>Code</Type>`
	Type = "Name"
	`<Value>Boston</Value>`
	`<Type>Name</Type>`
	Type = "Path"
	`<Value>All Locations\United States\Massachusetts\Boston</Value>`
	`<Type>Path</Type>`
	<Type> determine what type of value is passed in the value field to associate the correct category.
	<Type>Name</Type> can **not** be used if duplicate values exist within your category tree.

For more details see: Category Name/Value/Type syntax (see page 26)

Assignment categories

Assignment categories apply to users who provide some service or have viewing authority over employees benefiting from an event. Assignment categories are used for the automatic assignment of tasks based on category matching.

Employees entered in Onboarding for the sole purpose of benefiting from a life cycle event *do not require* assignment category values. Employees entered in Onboarding for the purpose of completing or viewing other employees tasks, *do use* assignment category values and/or team assignments to interact with other employee's tasks.

Assignment category values are not required elements. If you import an active user and intend for them to automatically be assigned tasks, team membership is required. If the employee will be automatically assigned category specific tasks, assignment categories must be added to the user. You can edit their profile through Onboarding to add assignment categories or you may include team membership and assignment categories in the XML import record.

The child category syntax is the same for assignment categories and event categories. The parent category syntax for assignment categories is the <Categories> element in the <user> record.

Here's sample syntax for assignment categories:

```
<!-- Note: category notation for assignment categories use the "abstract
category" (name or code) as available from the Export record. -->
   <Categories>
    <Category>
     <Name>Department</Name>
     <Value>CUSTSERV</Value>
     <Type>Code</Type>
    </Category>
    <Category>
     <Name>Job Type</Name>
     <Value>501ENG</Value>
     <Type>Code</Type>
    </Category>
    <Category>
     <Name>Location</Name>
     <Value>All Locations\United States\Massachusetts\Bedford</ Value>
     <Type>Path</Type>
     </Category>
   </Categories>
```

A user with an event

When creating a new employee in Onboarding, you can create an event for the employee (as the example above). There are syntax rules for including an event with your user record:

- All fields in the event definition are required in the record. Event categories are required if you are including an event.
- The naming convention of the XML tags is derived from the fields defined in the event definition (in the Onboarding user interface).
    - Each defined *of the defined categories* use Name/Value/Type syntax.
    - Each defined *person and date fields* use Name/Value pair syntax
    - The record must contain valid event field values as seen through the Onboarding user interface.

Note: Event categories <Name> references the category alias.

Example required fields of the default Onboarding event are:

```
<Event>
 <Name>Onboarding</Name>
 <Person>
```

```
    <Name>HR Coordinator</Name>
    <Value>Jenny.Rogers@silkroadtech.com</Value>
   </Person>
   <Person>
      <Name>Manager</Name>
      <EmployeeID>
          <ID>888</ID>
          <EmployeeIDType>Employee_HRISID</EmployeeIDType>
      </EmployeeID>
  </Person>
   <Date>
    <Name>Start</Name>
    <Value>2008-05-12</Value>
   </Date>
   <Category>
    <Name>Department</Name>
    <Value>Customer Service</Value>
    <Type>Name</Type>
   </Category>
   <Category>
    <Name>Job Type</Name>
    <Value>Staff</Value>
    <Type>Name</Type>
   </Category>
   <Category>
    <Name>Location</Name>
    <Value>East</Value>
    <Type>Name</Type>
   </Category>
  </Event>
```

A user with a key properties

You can choose to include key property values for the employee (as the example above). Key properties are employee attributes available as search and filter criteria through the user interface. Key properties have one value for each field type.

The child category syntax is similar to event category syntax. It includes three parts: category, person, and date. As with the event categories, key properties use the category alias. The parent node syntax for key property categories is:

- the **<KeyProperties>** element in the <user> record

Key properties (categories, people, and/or dates) can be established on the record without being pre-established through the Manage Key Properties page.

A convenient reference page is the Administration>Localization page. This page deals with multi-lingual labels, but it is a useful as a centralized reference for stored codes. For example, any code listed as type: category alias, date label, or person title can be referenced as a key property to add to an employee.

> Note: Category notation for key property categories use the "category alias" (name or code) as available for reference from the Localization page

Here's an example of syntax for a user key properties (one category, two people, and one date) for a single user:

```
<!-- Note: category notation for key property categories use the "category alias" (name or code) as available for reference from the Localization page -->
<KeyProperties>
  <Category>
   <Name>Current Location</Name>
   <Value>EAST</Value>
   <Type>Code</Type>
  </Category>
  <Person>
   <Name>HR Coordinator</Name>
   <Value>RC2.HRCoord</Value>
  </Person>
  <Person>
    <Name>Manager</Name>
    <EmployeeID>
        <ID>888</ID>
        <EmployeeIDType>Employee_HRISID</EmployeeIDType>
    </EmployeeID>
</Person>
  <Date>
   <Name>Distribution Date</Name>
   <Value>2009-09-02</Value>
  </Date>
</KeyProperties>
```

Shared dashboards

If you are importing an active employee, you can choose to give the new employee access to another employee's dashboard by establishing them as an alternate employee on a shared dashboard.

Similar to the bulk import syntax for events, the specific syntax used to create the category criteria used to filter a shared dashboard are configured based on your company's Onboarding implementation. The XML syntax is based on what you have defined in your implementation. You should cross reference the Onboarding Manage Shared Dashboard pages to understand the valid values of the defined categories.

Note: If the <SharedDashboards> node is included in the employee import syntax, the user executing the script must be a member of a team assigned the "Manage Shared Dashboards" privilege.

The syntax rules for SharedDashboards are:

1. Within the <SharedDashboard**s**> parent node, each Shared Dashboard is defined as a child node
   - <SharedDashboard></SharedDashboard>

2. The employee(s) who will be able to view the shared dashboard of the new employee are identified with *one* of the following unique dentures. *One* of the following attributes of the SharedWithEmployee_ is required:
   - SharedWithEmployee_GUID
   - SharedWithEmployee_Email
   - SharedWithEmployee_AuthParam
   - SharedWithEmployee_LoginID
3. Three options are available regarding the activation of the Shared Dashboard:
   - immediate activation for a non specific date range, specify:

```
<Activated>True</Activated>
```

- for a date specific activation specify a start and end date in YYYY-MM-DD format:

```
<ActivationFromDate>2009-04-21</ActivationFromDate>
<ActivationToDate>2009-04-28</ActivationToDate>
```

- to establish a shared dashboard but not as currently activated, specify:

```
 <Activated>False</Activated>
```

4. A shared dashboard can be delegated by one or more specific categories. The category syntax follow the category syntax used for assignment categories. Each category is coupled by three child elements:

```
<Name> </Name>
<Value> </Value>
<Type> </Type>
```

5. An error will occur on the SharedDashboard node for the following reasons:
   - the SharedWithEmployee_* cannot be found, categories or category values are invalid,
   - ActivationFromDate, ActivationToDate have invalid format, or do not form a valid range
   - Activated, ActivationFromDate, ActivationToDate conflict with each other

The following syntax activates a shared dashboard to allow the new employee to view a dashboard for two other employees:

```
<SharedDashboards>
  <SharedDashboard>
    <SharedWithEmployee_LoginID>Jenny.Jones</ SharedWithEmployee_LoginID>
    <ActivationFromDate>2009-04-21</ActivationFromDate>
    <ActivationToDate>2009-07-21</ActivationToDate>
    <Category>
      <Name>Location</Name>
      <Value>Bedford_MA</Value>
      <Type>Code</Type>
    </Category>
    <ForwardNotifications>All</ForwardNotifications>
  </SharedDashboard>
  <SharedDashboard>
    <SharedWithEmployee_Email>Roger.Lee@some.com</ SharedWithEmployee_Email>
    <Activation>True</Activation>
    <ForwardNotifications>None</ForwardNotifications>
  </SharedDashboard>
</SharedDashboards>
```

Employee elements

Each sub-child element of the <user> record are described below. Active users require the all elements to be populated with valid values. Employees can be created as pending or active users.

- Pending user status can apply to any employee
- New Employees created as active users require the <ForceActiveNewHire> value of 1 (one).

- If an event is created with a *pending* employee record, corresponding tasks are created in the Review and Approve process. A pending user with an event has an extra "Review *eventname* event" command button at the bottom of the form.
- If an event is created with a *active* employee record, the tasks are created as part of successfully creating the user.
- *One* event can be created for a new employee through the bulk import. An event can be created in a new employee import. Events can not be created through an XML bulk edit.
- To customize the tasks created for an imported employee, create the user as a pending user. In the Review event form, choose to customize the task assignment.

Basic Employee Profile

Node	Required (for an Active Employee)	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	✓	XML Document header	
<users>	✓	Parent Node occurs exactly once	
<user>	✓	Node for each user, can occur many times	
<ForceActiveNewHire> </ForceActiveNewHire>		Identifies if the user should be an active or pending user.	Valid values are: 1 (one) - create an active user 0 (zero) - default - create a pending user. Setting the value to 1 to create an active new employee will: 1. Create an active new employee who can immediately log in to Onboarding 2. Send the corresponding emails associated with new employee. 3. If an event is included in the XML record, the event will launch simultaneously to employee creation.

Node	Required (for an Active Employee)	Description	Notes
<ApprovalManager> </ApprovalManager>		Optional for pending user Not applicable to active users	Identifies the person assigned to review the pending employee information in the Onboarding user interface Pending Employee List. The specific person can be identified by one of these unique identifiers: • login id • email address* • user GUID *If email address is not unique it is not a valid identifier Example: <ApprovalManager>George.Rogers</ApprovalManager>
First_Name	✓	Optional but recommended to create a pending user	
Middle_Initial			
Last_Name	✓	Optional but recommended to create a pending user	
HireDate			Expected format is YYYY-MM-DD. HireDate is referenced by rules that apply to your company's legal obligation to retain I-9 forms on file.
TerminationDate			Expected format is YYYY-MM-DD TerminationDate is referenced by rules that apply to your company's legal obligation to retain I-9 forms on file. Example: <TerminationDate>2008-11-23</TerminationDate>

Node	Required (for an Active Employee)	Description	Notes
Email	⚠	Customer configuration can specify this as a required field. Value is the new user's email address.	Example element syntax: <Email>Sally.O@silkroa[1]dtech.com</E[2]mail>
LoginID		The value that a user will use to login to Onboarding	If LoginID is omitted, Onboarding will generate a default value of First_Name + "." + Last_Name. Example element syntax: <LoginID>Sally.Obrien</LoginID>
Password		Value is the new user's Onboarding assigned Password. **Applies only to an employee created as an Active user.** Password is an applicable element only if <AuthType> is set to standard.	Example element syntax: <Password>!Welcome$</Password> Note: If <Password> is not included in the record, and <ForceActiveNewHire>1</ForceActiveNewHire>, a password will be automatically generated and sent to the email address provided in the <Email> value. NOTE: it is important to include a valid <Email> value with this option allowing the employee to receive the Welcome email.
ForcePassword Change		Option to require a new employee to change their password the first time they log in. They will be prompted for their original password, a new password and a confirmation of the new password.	Valid values are: 1 - Employee will be forced to modify password at first login 0 - (default) Employee will be not be prompted to modify password at first log in Example Syntax:<ForcePasswordChange>1</ForcePasswordChange>

1 mailto:Sally.O@silkroadtech.com
2 mailto:Sally.O@silkroadtech.com

Node	Required (for an Active Employee)	Description	Notes
AuthType	⚠	Default = "standard" Required if External Authentication (CUProfile) Optional for Standard	Valid values are: standard - user will be authenticated by Onboarding CUProfile - user will be authenticated by an application that is external to Onboarding Example element syntax: <AuthType>CUProfile</AuthType> Used in conjunction with: <AuthParam>UniqueUID</AuthParam> Note: By default, pending users are created as standard users.
SSOAuthParam AuthParam	⚠	Required if External Authentication	NOTE: Either node is accepted. AuthParam node was updated to SSOAuthParam. SSOAuthParam is a required element if AuthType is CUProfile. Valid value is the new user's unique in the authentication system. Used in conjunction with: <AuthType>CUProfile</AuthType> Example element syntax: <SSOAuthParam>jjackson</SSOAuthParam> OR <AuthParam>jjackson</AuthParam>
Employee_HRISID	⚠	Customer configuration can specify this as a required field.	Must be unique across Onboarding users. This field exists to cross reference an employee with their ID in another (external corporate) system. Employee_HRISID is exposed in the user interface if a label is defined in the configuration. This field can be configured to be optional or required. Example element syntax: <Employee_HRISID>2012001</Employee_HRISID>
PhoneWork			

Node	Required (for an Active Employee)	Description	Notes
MobileCountryCode			Must have SMS enabled to update field. Required if profile is being opted in to receive text messages. Must provide the 1,2 or 3 digit Country Code Codes: https://countrycode.org/ Example element syntax: <MobileCountryCode>1</MobileCountryCode>
PhoneMobile			When SMS is enabled the Mobile number must be added without Country Code
TimeZone			Must have SMS enabled to update field. Required if profile is being opted in to receive text messages. Valid values are: Allowed Time Zones Example Syntax:<TimeZone>1</TimeZone>
SmsOptIn			Must have SMS enabled to update field. Valid values are: 1 - Employee will be opted into receiving text messages 0 - Employee will be opted out of receiving text messages Example Syntax:<TimeZone>Eastern Standard Time</TimeZone>

Node	Required (for an Active Employee)	Description	Notes
Custom fields from the employee profile	⚠	Required based on custom implementation	The XML (or form) field name directly corresponds to the field name entered in the content element editor (no prefix). If a custom field is created as required, it is required to be populated to successfully import an *active* employee. If the imported employee is a pending user, the required custom fields are required for the approval of a *pending* employee. If a custom field is a selection type field (drop down list, radio button, or checkbox) the valid value is a defined field value (not a display value). The field values are case sensitive. Example element syntax: <Gender>M</Gender>
Any additional nodes from the sections below.			
</user>	✓	Close each user, can occur many times	
</users>	✓	Close Parent Node occurs exactly once	

Team Membership

Team elements are optional when creating an employee.

Node	Description	Notes
<Teams>	Parent element for team operations.	

Node	Description	Notes
<Team>	Team assignment adds the employee as a team member.	Team value is defined in the Manage Teams page of the Onboarding user interface. **Example Syntax** `<Team>Managers</Team>`
<TeamController>	Team Controller adds the employee as a controller of a team. A team controller can add and remove other team members from the team(s) they are controllers.	Team value is defined in the Manage Teams page of the Onboarding user interface. **Example Syntax** `<TeamController>Managers</TeamController>`
</Teams>		

Assignment Categories

Assignment category elements are optional when creating an employee.

Node	Description	Notes
<Categories>	Parent element for Assignment categories.	Assignment categories apply to employees who will be assigned tasks or view other's tasks. Note: See Category Syntax notes in the Event Definition section

Node	Description	Notes

<Category>
<Name|Code>
<Value>
<Type>
</Category>
	Occurs for each category defined.
	The Category can be looked up by name in the default language OR code. The element changes depending how the category will be identified.
	Valid names are text as it is entered in the "Category Name" field of the manage key properties page.
	Valid codes are available on the manage localization page.
	For more details on the category syntax see: Category Name/ Value/Type syntax (see page 26).
		Example Syntax: Name
		`<Category>`
		`<Name>Work Location</Name>`
		`<Value>MA_BEDFORD</Value>`
		`<Type>Code</Type>`
		`</Category>`
		Example Syntax: Code
		`<Category>`
		`<Code>Cat_Work_Loc</Code>`
		`<Value>MA_BEDFORD</Value>`
		`<Type>Code</Type>`
		`</Category>`

</Categories>		

E-Verify

E-Verify elements are optional when creating an employee.

Node	Description	Notes
<eVerifyCaseNumber>	eVerifyStatus must also be populated if eVerifyCaseNumber is populated.	If you are importing an employee already processed through e-Verify, you can import the DHS Case Number and Status for display on the Employee Profile - I-9 tab and I-9 Dashboard. DHS Case Number generated by an external system executing eVerify methods. **Example Syntax** `<eVerifyCaseNumber>23435632</eVerifyCaseNumber>`

Node	Description	Notes
<eVerifyStatus>	eVerifyCaseNumber must also be populated if eVerifyStatus is populated.	If you are importing an employee already processed through e-Verify, you can import the DHS Case Number and Status for display on the Employee Profile - I-9 tab and I-9 Dashboard. Valid eVerify status codes are: • "IQ" - Invalid Query • "SELFT" - Self-Terminated • "EUAUTH" - Resolved Unauthorized / Terminated • "SNTERM" - Employee Not Terminated • "EAUTH" - Resolved Authorized **Example syntax for an Authorized employee:** `<eVerifyStatus>EAUTH</eVerifyStatus>`
<eVerifyDate>	eVerifyDate will default to today's date if it is not inlcuded in the recored with eVerifyCaseNumber and eVerifyStatus	If you are importing an employee already processed through e-Verify, you may import the DHS Case Number and Status for display on the Employee Profile - I-9 tab and I-9 Dashboard. eVerify Case date. Expected format is YYYY-MM-DD. **Example Syntax** `<eVerifyDate>2008-11-23</eVerifyDate>`

Event

You can create an event when you create the new employee. Events are configured based on your company's implementation of Onboarding. The XML syntax is based on what you have defined in your implementation. You should cross reference the Onboarding Event Definition pages to understand the elements of your event definition and each element's accepted values.

It is useful to familiarize yourself with the following definitions in the Onboarding user interface to map your XML syntax.

- **Administration>Define Events** to verify the fields defined in your events, categories, and category alias names.
- **Administration>Manage Categories** to examine the expected category names and values and the hierarchy Onboarding expects.

1. In Onboarding, under **Define Events**, for each event each itemized Category Name, People Name, or Dates are recognized by the import process as required values. If you either omit any of these items or, if your record includes an invalid value, the record will fail.
2. Each itemized Category Name, People Name, and Date Name is the "Name" value for the XML syntax.
    - The node for people is <Person> </Person>

Note: An event can be added to a new employee during a bulk import. An event can added to an existing employee through the UpdateEvent (see page 174) method.

Node	Required (for an Active Employee with an event)	Description	Notes
<Event>	⚠	<Event> is a parent element. When it is included all the elements required to define the event are required.	Occurs one time for one event. One event is allowed per user.
<Name>	✓	Valid values are text as it is entered in the "Event Name" field of the Manage Events page.	Child element of <Event> **Example Syntax** `<Name>Leave of Absence</Name>`

<Category>
<Name|Code>
<Value>
<Type>
</Category>
	✓	Occurs for each category defined on the event
		The Category can be looked up by name in the default language OR code. The element changes depending how the category will be identified.
		Valid names are text as it is entered in the "Category Name" field of the Manage Events page.
		Valid codes are available on the manage localization page.
		For more details on the category syntax see: Category Name/Value/Type syntax (see page 26).
			Example Syntax: Name
			`<Category>`
			`<Name>Work Location</Name>`
			`<Value>MA_BEDFORD</Value>`
			`<Type>Code</Type>`
			`</Category>`
			Example Syntax: Code
			`<Category>`
			`<Code>Cat_Work_Loc</Code>`
			`<Value>MA_BEDFORD</Value>`
			`<Type>Code</Type>`
			`</Category>`

Node	Required (for an Active Employee with an event)	Description	Notes

<Person>
<Name|Code>
<Value>
</Person>
	✓	Occurs for each category defined on the event
		The Person can be looked up by name in the default language OR code. The element changes depending how the person will be identified.
		Valid names are text as it is entered in the "Person Name" field of the Manage Events page.
		Valid codes are available on the manage localization page.
			Example Syntax: Name
			`<Date>`
			`<Name>HR Coordinator</Name>`
			`<Value>George.Rogers</Value>`
			`</Date>`
			Example Syntax: Code
			`<Person>`
			`<Code>Person_2</Code>`
			`<Value>George.Rogers</Value>`
			`</Person>`

<Date>
<Name|Code>
<Value>
</Date>
	✓	Occurs one time for each "date name" defined on the event.
		The Date can be looked up by name in the default language OR code. The element changes depending how the date will be identified.
		Valid names are text as it is entered in the "Date Name" field of the Manage Events page.
		Valid codes are available on the manage localization page.
		Expected format for the value is YYYY-MM-DD
			Example Syntax: Name
			`<Date>`
			`<Name>Start</Name>`
			`<Value>2008-11-23</Value>`
			`</Date>`
			Example Syntax: Code
			`<Date>`
			`<Code>Date_1</Code>`
			`<Value>2008-11-23</Value>`
			`</Date>`

</Event>			

Key properties

You can optionally include key properties when you create a new employee. Key Property syntax mimics the syntax used when adding an event in that the same attributes are collected: category, person, and date. All of the same syntax rules apply.

The following rules apply for key properties:

- The parent node is <KeyProperties> </KeyProperties>
- The <Category> <Name> <Value> <Type> child node refers to a category alias.
- You may include as many <Category> <Person> or <Date> fields as you like.

Node	Description	Notes
<KeyProperties>	<KeyProperties> is a parent element. When it is included all the elements required to define the event are required.	

<Category>
<Name|Code>
<Value>
<Type>
</Category>
	Occurs for each category defined on the on the manage key properties page.
	The Category can be looked up by name in the default language OR code. The element changes depending how the category will be identified.
	Valid names are text as it is entered in the "Category Name" field of the manage key properties page.
	Valid codes are available on the manage localization page.
	For more details on the category syntax see: Category Name/ Value/Type syntax (see page 26).
		Example Syntax: Name
		`<Category>`
		`<Name>Work Location</Name>`
		`<Value>MA_BEDFORD</Value>`
		`<Type>Code</Type>`
		`</Category>`
		Example Syntax: Code
		`<Category>`
		`<Code>Cat_Work_Loc</Code>`
		`<Value>MA_BEDFORD</Value>`
		`<Type>Code</Type>`
		`</Category>`

Node	Description	Notes

<Person>
<Name|Code>
<Value>
</Person>
	Occurs for each category defined on the manage key properties page.
	The Person can be looked up by name in the default language OR code. The element changes depending how the person will be identified.
	Valid names are text as it is entered in the "Person Name" field of the manage key properties page.
	Valid codes are available on the manage localization page.
		Example Syntax: Name
		`<Date>`
		`<Name>HR Coordinator</Name>`
		`<Value>George.Rogers</Value>`
		`</Date>`
		Example Syntax: Code
		`<Person>`
		`<Code>Person_2</Code>`
		`<Value>George.Rogers</Value>`
		`</Person>`

<Date>
<Name|Code>
<Value>
</Date>
	Occurs one time for each "date name" defined on the manage key properties page.
	The Date can be looked up by name in the default language OR code. The element changes depending how the date will be identified.
	Valid names are text as it is entered in the "Date Name" field of the manage key properties page.
	Valid codes are available on the manage localization page.
	Expected format for the value is YYYY-MM-DD
		Example Syntax: Name
		`<Date>`
		`<Name>Start</Name>`
		`<Value>2008-11-23</Value>`
		`</Date>`
		Example Syntax: Code
		`<Date>`
		`<Code>Date_1</Code>`
		`<Value>2008-11-23</Value>`
		`</Date>`

</KeyProperties>		

Shared dashboards

Establishing shared dashboards is optional when creating an employee.

Node	Required (to setup shared dashboard)	Description	Notes
<SharedDashboards>	⚠	<SharedDashboards> is a parent element. When it is included all the elements required to define shared dashboards are required.	Occurs one time for each new active employee. <SharedDashboards> on pending employees will be ignored.
<SharedDashboard>	✓		

<SharedWithEmployee_*LoginID*>	✓	Unique identifier for the employee dashboard to be accessed.	Optional identifiers are:
			• SharedWithEmployee_GUID
			• SharedWithEmployee_Email
			• SharedWithEmployee_AuthParam
			One identifier is required for each <SharedDashboard>
			Example Syntax
			`<SharedWithEmployee_LoginID>Jenny.Jones</SharedWithEmployee_LoginID>`
			`<SharedWithEmployee_GUID>000392e2-9a5c-4ceb-ab60b95a20e72c30</SharedWithEmployee_GUID>`
			`<SharedWithEmployee_Email>Jenny.Jones@some.com</SharedWithEmployee_Email>`
			`<SharedWithEmployee_AuthParam>JJones246</SharedWithEmployee_AuthParam>`

<Activation>	⚠	Required only if an activation date range is not specified. It can be activated through the Share Dashboard page at a future date.	Possible values: • True - activate the shared dashboard immediately and indefinitely • False - Shared Dashboard entry is created but not activated. **Example Syntax** `<Activation>True</Activation>`

Node	Required (to setup shared dashboard)	Description	Notes
ActivationFromDate		Shared Dashboard activation start date.	Expected format is YYYY-MM-DD. **Example Syntax** `<ActivationFromDate>2009-04-21</ActivationFromDate>`
ActivationToDate		Shared Dashboard activation end date.	Expected format is YYYY-MM-DD. **Example Syntax** `<ActivationToDate>2009-04-28</ActivationToDate>`
<Category> <Name|Code> <Value> <Type> </Category>	✓	Occurs for each category that will be used to limit the scope of the shared dashboard. The Category can be looked up by name in the default language OR code. The element changes depending how the category will be identified. Valid names are text as it is entered in the "Category Name" field of the manage localization page. Valid codes are available on the manage localization page. For more details on the category syntax see: Category Name/Value/Type syntax (see page 26).	**Example Syntax: Name** `<Category> <Name>Work Location</Name> <Value>MA_BEDFORD</Value> <Type>Code</Type> </Category>` **Example Syntax: Code** `<Category> <Code>Cat_Work_Loc</Code> <Value>MA_BEDFORD</Value> <Type>Code</Type> </Category>`

Node	Required (to setup shared dashboard)	Description	Notes
<ForwardNotifications>		Optionally identified which notifications will be forwarded to the recipient of the shared dashboard. Default is All Example syntax	Possible values are: • "All" - All notifications related to the shared dashboard tasks will be forwarded to the alternate (new employee) • "None"- None of the notifications related to the shared dashboard tasks will be forwarded to the alternate (new employee) • "WarningsOverdue" - Notifications related only Warnings or Overdue notices related to the shared dashboard tasks will be forwarded to the alternate (new employee) **Example Syntax** <ForwardNotifications>All</ForwardNotifications>
</SharedDashboard>	✓		
</SharedDashboards>	✓		

Example XML Bulk User import

```
<?xml version="1.0"?>
 <users>
  <user>
    <First_Name>James</First_Name>
    <Middle_Name>Jay</Middle_Name>
    <Last_Name>Walters</Last_Name>
    <LoginID>James.Walters</LoginID>
    <Email>James.Walters@silkroadtech.com</Email>
    <HireDate>2008-08-12</HireDate>
    <ForceActiveNewHire>1</ForceActiveNewHire>
    <AuthType>CUProfile</AuthType>
    <SSOAuthParam>James.Walters</SSOAuthParam>
    <Employee_HRISID>99221341</Employee_HRISID>
```

```
    <Phone_Work>317-555-1234</Phone_Work>
    <Phone_Mobile>231-555-6543</Phone_Mobile>
<!-- Note: category notation for assignment categories use the "abstract
category" (name or code) as available from the Export record. -->
   <Categories>
    <Category>
     <Name>Department</Name>
     <Value>CUSTSERV</Value>
     <Type>Code</Type>
    </Category>
    <Category>
     <Name>Job Type</Name>
     <Value>501ENG</Value>
     <Type>Code</Type>
    </Category>
    <Category>
     <Name>Location</Name>
     <Value>All Locations\United States\Massachusetts\Bedford</ Value>
     <Type>Path</Type>
     </Category>
   </Categories>
   <ForceActiveNewHire>1</ForceActiveNewHire>
   <Password>!secret!</Password>
   <Teams>
    <Team>TeleSupport</Team>
    <Team>EastCoastSupervisors</Team>
    <TeamController>EastCoastSupervisors</TeamController>
   </Teams>
<!-- Note: category notation for key property categories use the "category alias" (name
or code) as available for reference from the Localization page -->
   <KeyProperties>
    <Category>
     <Name>Current Location</Name>
     <Value>EAST</Value>
     <Type>Code</Type>
    </Category>
    <Person>
     <Name>HR Coordinator</Name>
     <Value>RC2.HRCoord</Value>
    </Person>
    <Person>
     <Name>Coach</Name>
     <Value>Shelly.Sharem</Value>
    </Person>
    <Date>
     <Name>Distribution Date</Name>
     <Value>2009-09-02</Value>
    </Date>
   </KeyProperties>
   <Event>
    <Name>Onboarding</Name>
```

```
      <Person>
       <Name>HR Coordinator</Name>
       <Value>hwise@Onboarding.com</Value>
      </Person>
      <Person>
       <Name>Manager</Name>
       <Value>mtucker@Onboarding.com</Value>
      </Person>
      <Date>
       <Name>Start</Name>
       <Value>2008-08-07</Value>
      </Date>
<!-- Note: category notation for event categories use the "category alias" (name or
code) as available for reference from the Localization page -->
      <Category>
       <Name>Department</Name>
       <Value>Engineering</Value>
       <Type>Name</Type>
      </Category>
      <Category>
       <Name>Job Type</Name>
       <Value>Engineer</Value>
       <Type>Name</Type>
      </Category>
      <Category>
       <Name>Location</Name>
       <Value>All Locations\United States\Massachusetts\Bedford</ Value>
       <Type>Path</Type>
      </Category>
    </Event>
<!-- SharedDashboards nodes -->
    <SharedDashboards>
     <SharedDashboard>
      <SharedWithEmployee_LoginID>Jenny.Jones</ SharedWithEmployee_LoginID>
      <ActivationFromDate>2009-04-21</ActivationFromDate>
      <ActivationToDate>2009-07-21</ActivationToDate>
      <Category>
        <Name>Location</Name>
        <Value>Bedford_MA</Value>
        <Type>Code</Type>
      </Category>
      <ForwardNotifications>All</ForwardNotifications>
    </SharedDashboard>
    <SharedDashboard>
      <SharedWithEmployee_Email>Roger.Lee@some.com</ SharedWithEmployee_Email>
      <Activation>True</Activation>
      <ForwardNotifications>None</ForwardNotifications>
     </SharedDashboard>
    </SharedDashboards>
<!-- E-Verify case fields -->
    <eVerifyCaseNumber>23435632</eVerifyCaseNumber>
```

```
  <eVerifyStatus>EAUTH</eVerifyStatus>
  <eVerifyDate>2008-11-23</eVerifyDate>
 </user>
</users>
```

Bulk Import error codes

If an error exists, the output includes a <UserError> tag. For example:

```
<?xml version="1.0"?>
<BulkImportResults>
 <Note>1 active users created</Note>
 <Note>0 pending users created</Note>
 <Note>Parsing started at Thursday, July 24, 2008 3:14:25 PM and ended at Thursday, July 24, 2008 3:14:25 PM</Note>
 <Note>Total duration: .0019 seconds (.0001 seconds were spent creating users)</Note>
 <CreatedActiveUsers>
  <CreatedActiveUser>John Smith</CreatedActiveUser>
   <Employee_HRISID />
   <SSOAuthParam />
   <Guid>cb9424d1-d9d5-4112-868c3ca6423ae1ed</Guid>
   <LoginID>John.Smith</LoginID>
   <Email>jsmith@MyCompany.com</Email>
   <LifeSuiteID></LifeSuiteID>
   <EventID>46</EventID> <!-- if an event was launched on upload -->
   <FullName>John Smith</FullName>
  </CreatedActiveUsers>
  <UserError>
   <Name>Jane(User's e-mail: Jane.Brown02@silkroadtech.com)</ Name>
   <Error>
    <Type>User Save Error</Type>
    <Message>User marked as ForceActiveNewHire, but does not have required 'Last_Name' data</Message>
    <ErrorCode>30052</ErrorCode>
   </Error>
  </UserError>
 </BulkImportResults>
```

Each Bulk Import error is communicated with the following elements:

```
<UserError>
 <Error>
  <Type></Type>
  <Message></Message>
  <ErrorCode></ErrorCode>
 </Error>
</UserError>
```

BulkUserUpload error codes

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The SSOAuthParam value is already in use by another user	21022	Attempt to add a user when specified AuthParam or SSOAuthParam is used by an existing user
Invalid Input	The Employee_HRISID is already in use by another user	20125	Attempt to add a user when specified Employee_HRISID is used by an existing user
Invalid Input	The LifeSuiteID value is already in use by another user. Attempt to add a user when specified LifeSuiteID is used by an existing user	20126	RESERVED For SilkRoad
Event Already Exists	X event already exists for user:Y	30000	One "In Progress" event (per event type) at a time.
Fatal Error	Fatal Error	30001	Caused by malformed XML
Fatal Error	Failed to add event to user because the sessionid executing the method is executing the method is	30002	Not launched by the event coordinator
Fatal Error	Unknown event name	30003	
Invalid Input	Invalid category value name for *named category:value*	30004	The value in passed for <Type>Name</ Type> did not match a valid category name.
Invalid Input	Invalid category value name for *named category:value*	30005	The value in passed for <Type>Code</ Type> did not match a valid category Code.
Invalid Input	Invalid category value name for *named category:value*	30006	The value in passed for <Type>Path</ Type> did not match a valid category Path.
Event Error	Named category does not exist on the Event definition	30007	Event values launched for imported user must map to field defined on the event definition

<Type>	<Message>	<ErrorCode>	Description
Event Error	Named person does not exist on the Event definition	30008	
Invalid Input	Missing abstract category node under category node	30015	The <Category> node is missing the <Name> child node.
Invalid Input	Missing category value node under category node	30016	The <Category> node is missing the <Value> child node
Invalid Input	Missing category value node under category node	30017	The <Category> node is missing the <Type> child node
Invalid Input	Unknown person (manager) name	30018	
Invalid Input	Named person does not exist	30019	
Invalid Input	Failed to update event because the sessionid executing the method is not an event coordinator	30020	
Invalid Input	Unknown date name	30021	
Invalid Input	Named date does not exist on the event definition	30022	
Invalid Input	Date format is invalid	30023	Expected date format is YYYY-MM-DD
Invalid Input	More than one category value name found for value category: *category name*	30024	<Type>Name</Type> was specified but Name is not a unique category identifier
Invalid Input	No user identification node could be found for the {0} event node	30025	
Invalid Input	Missing person (manager) name	30026	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Missing person (manager) value	30027	Applies to all missing <Person> nodes. The specific <Name/> node is not specified. .
Invalid Input	Missing date name	30028	
Invalid Input	Missing date value	30029	
Invalid Input	Missing category name node under category node	30030	Each <Category> occurence must have a <Name> node
Invalid Input	Invalid category value type (Type entered: *x*) must be 'Name', 'Code', or 'Path'	30031	Valid <Type> values are: • <Type>Name</Type> • <Type>Code</Type> • <Type>Path</Type>
Invalid Input	Invalid event name '*x*' found	30032	Undefined <Event><Name> value was specified.
Invalid Input	Parsing failed due to: *x*	30033	Unrecognized node specified.
Invalid Input	Unknown date name: *x*	30034	Undefined <Date> value specified
Invalid Input	Invalid manager identifier entered for *x* manager: {1}	30035	Undefined manager name specified on event or unidentified ApprovalManager specified on a pending user
Invalid Input	Invalid manager name entered: {0}");	30036	Invalid <Value> for the coorsponding <Person> <Name>
Invalid Input	Invalid date value entered for *x* date, must be in YYYY-MM-DD format: {1}	30037	
Invalid Input	User marked as ForceActiveNewHire, but does not have required *x data*	30038	Required fields must be included in the ForceActiveNewHire record.

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Missing category name node under category node	30039	
Invalid Input	The EmployeeIDType {0} entered for {1} is unknown. Please use one of the following: LifeSuiteID, Employee_HRISID, Email, LoginID, GUID, or SSOAuthParam.	30057	
Fatal Error	No users nodes found in file.	31000	No data.
Invalid Input	Termination Date not in proper YYYY-MM- DD format: *date*	31001	Invalid date format
Event Error	Users can be uploaded with 1 event at most. The *named* event must be removed.	31002	Bulk Import is restricted to one event per new user. Any additional events must be removed from the bulk import record. Additional events can be executed using the LaunchEvent method.
Event Error	Event node found but missing name	31003	<Event> tag specified without a <Name> tag
Event Error	Invalid event name *name* found	30003	Undefined Event name
Invalid Input	Unexpected *X* node found under *Y* node	31004	Malformed XML syntax where *X* and *Y are specified nodes.*
Invalid Input	Invalid role value: *value*	31006	Invalid team value specified
Invalid Input	Cannot create user from parsed data, despite the forceactivenewhire flag, because the user's data had errors	31007	Data error in user profile information.
User Validation	Ignoring unknown value: *key=value*	31008	Invalid node syntax

<Type>	<Message>	<ErrorCode>	Description
User Validation	Failed to create active user due to errors in the user information	31009	Data error in user profile information.
Event Error	Failed to create active user due to errors in the user event information	31010	
Invalid Input	Missing eVerify status data	31011	<eVerifyCaseNumber> and <eVerifyStatus> must both have values if either have values.
Invalid Input	Missing eVerify case number	31012	<eVerifyCaseNumber> and <eVerifyStatus> must both have values if either have values.
User Validation	message (exception)	31013	
User Validation	property : message	31014	
User Save Error	Failed to save a user due to a database error (message)	31015	Malformed XML caused database error.
User Save Error	Application exception while saving: message (Inner exception: inner message)	31016	Call SilkRoad support.
User Save Error	Exception while saving: *exc*	31017	
User Save Error	Failed to create pending user	31018	
User Save Error	The user invoking the bulk import does not have the privilege to create pending users.	31019	Incorrect privileges of the user executing the method

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The user invoking the bulk import does not have the privilege to create active users.	31035	Employee who obtained the SessionID must be on a team with the "Create User" privilege.
	XML user edit is stopping because an error was found and XMLUploadStopOnError is set to 1	31036	XMLUploadStopOnError set through the Administration ->Settings page. Set XMLUploadStopOnError to False to allow user edit to continue modifying valid records.
Invalid Input	The e-mail address (*x*) which was specified for the *n* user node is not used by any user in the system	31037	Invalid email address specified.
Invalid Input	The e-mail address (*x*) which was specified for the {*n*} user node is used by {*nn*} users so it cannot be used to uniquely identify a single user	31038	Non unique email address specified.
Invalid Input	An unkown error has occured	31039	
Invalid Input	Failed to load XML document (*x*)	31040	
Invalid Input	XML user edit failed due to:{*x*}	31041	
Invalid Input	No user could be found for the {*x*} user node	31042	
Invalid Input	Unkown abstract category: {*x*}	31043	Unknow "Alias Name" category reference specified.
Invalid Input	The EmployeeIDType {0} entered for {1} is unknown. Please use one of the following: LifeSuiteID, Employee_HRISID, Email, LoginID, GUID, or SSOAuthParam.	31053	EmployeeIdType and ID are both provided, and type is unknown, for a named person on a key property
Invalid Input	Insufficient Privileges to perform the action	34001	User executing script does not have the correct privilege (team membership or event coordinator designation) to execute the referenced action

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	{0} cannot be used because it is not a leaf category value (i.e. It has child category values)	36015	
Invalid Input	Key Property: Invalid Category Name - 'LOCATION'	39001	
Invalid Input	Key Property: Invalid Category Code - 'CATEGORY_1'	39002	
Invalid Input	Key Property: Invalid Person Name - 'MANAGER'	39003	
Invalid Input	Key Property: Invalid Person Code - 'PERSON_1'	39004	
Invalid Input	Key Property: Invalid Date Name - 'START'	39005	
Invalid Input	Key Property: Invalid Date Code - 'DATE_1'	39006	
Invalid Input	The following key properties are required but do not have a value: 'LIST OF MISSING REQUIRED KEY PROPERTIES'	39007	
Invalid Input	Key Property: Invalid category value name for 'LOCATION' category: 'FAKE TOWN'	39008	
Invalid Input	Key Property: Invalid category value path for ''LOCATION category: ''LOCATION category: 'FAKE/PATH/4U'	39009	
Invalid Input	Key Property: Invalid category value code for 'LOCATION' category: 'FAKE_CODE'	39010	
Invalid Input	Key Property: Invalid manager identifier entered for Manager manager:	39011	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Key Property: Date format is invalid	39012	Expected format is YYYY-MM-DD
Invalid Input	Missing ID for a named person	39014	
Invalid Input	Missing EmployeeIDType for a named person	39015	

Review and approve a pending user

Pending users overview

An important step after importing employees as pending users is to approve each pending user as an active user through the Onboarding user interface. Submitting a pending employee for approval established an employee as an active user.

- Active users can authenticated by the portal
- Active users can be located through the Find Employees page, allowing an event to be launched on their behalf
- Any event included with a pending employee is not activated until it is approved. No tasks are generated for a pending user until the user is submitted as an active employee.
- Pending employees can not be selected as providers until they are submitted as active employees.

Note: Employees imported through the file import as active employees do not appear in the Pending User List.

Activating pending users

Employees logging in to the Onboarding user interface who are members of a team with 'Pending Users" privileges have a menu option named "Pending Employees".

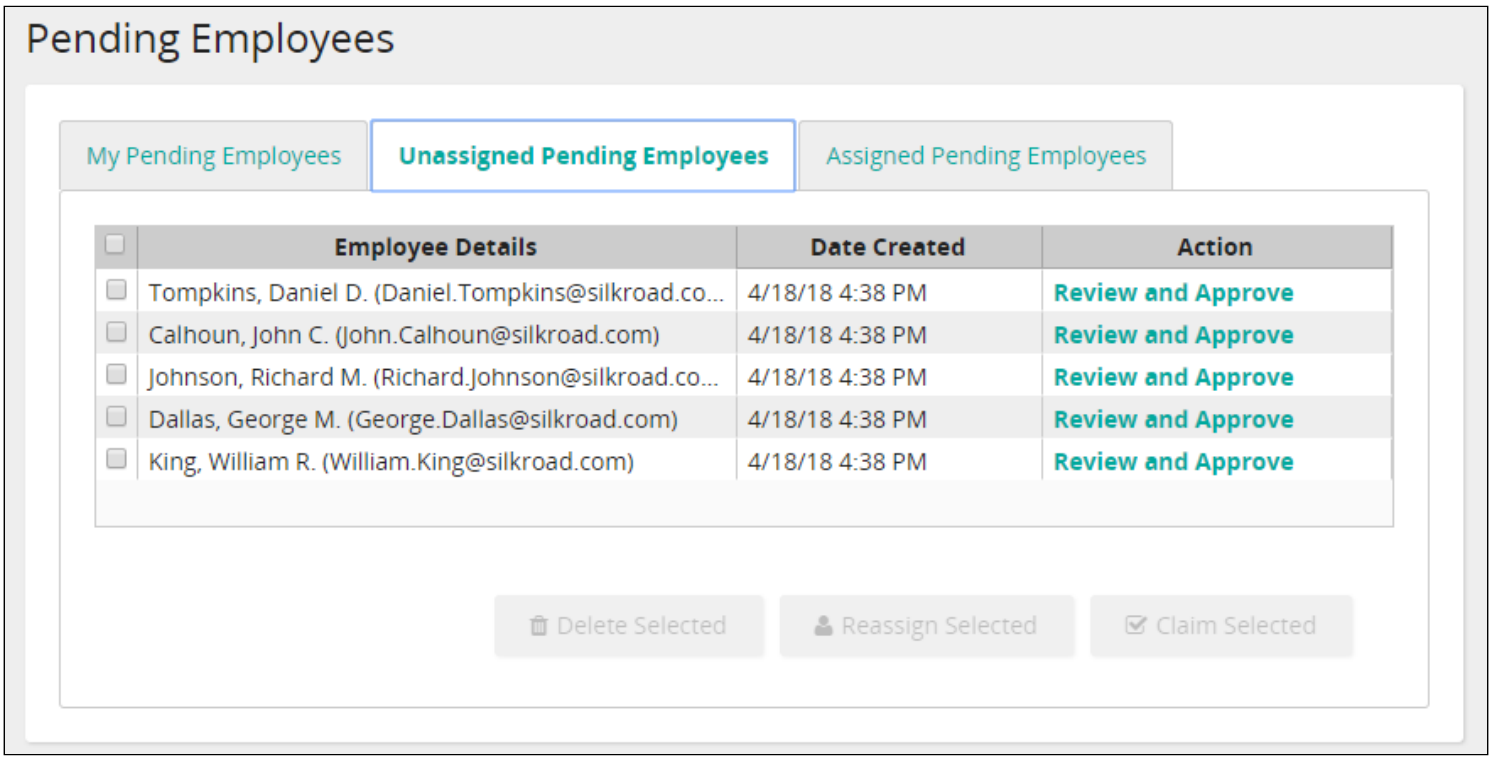
"Pending Employees" option to view the Pending Employees List is used to review and approve users imported into Onboarding with minimal information or information from an external system that needs review before the employee is activated.

The Pending Employee List can also be used as a staging area between other systems and Onboarding. Required employee information can be manually entered, reviewed, and/or approved before an active employee is created.

Pending Employee List management

Be default, all pending users are imported into the **Unassigned Pending Employees** list. To review, delete, claim, or assign pending users for approval, click on the Unassigned Pending Employees tab.

The pending employee can optionally be imported into a specific person's Assigned Pending Employee list. A specific pending user assignment can be specified by including the <ApprovalManager/> (see Chapter 1) node with the user in the import record.



Delete pending users

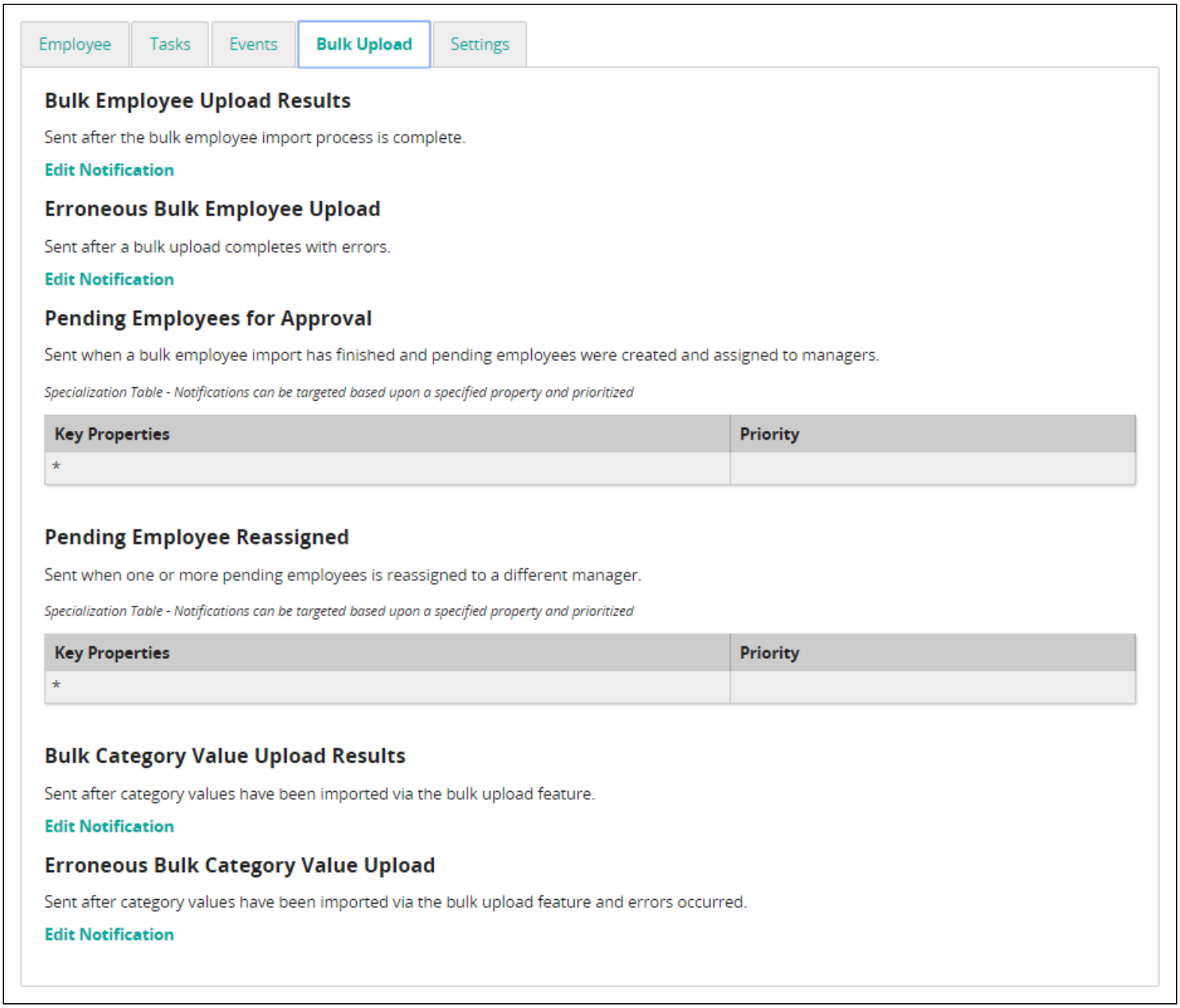
A Delete command is available to all assigned and unassigned pending users. Pending users have no tasks assigned to them and consequently can be deleted from RedCarpet with no impact.

Configure notification

Configuring success and failure email messages

You can configure bulk import notification messages in Onboarding.

The email message options are divided by successful and unsuccessful imports of each user record.



Users with privileges to edit Notifications can select the **Administration>Notifications** menu option. From the Notifications page, you can choose a notification message type for the bulk import process.

Turn off success (or error) email messages

Success (or error) email messages for a notification can be turned off by unchecking the Send Emails option for that notification.

XMLUserEdit

An XML record format is also available to update employee information such as employee profile information, team membership, team controller status, and assignment categories. You can also update an employee from standard authentication to external authentication or change an active employee to retired, or unretire a retired employee.

Editable employee fields of an active user can be updated via a post of an XML record, passing the XML to the available web method, or a file through the Onboarding user interface pages.

By default Onboarding employees have access to the Onboarding user interface. SilkRoad Support can explicitly remove access to the Onboarding interface upon request. Denying access to the Onboarding interface prohibits an employee created in Onboarding (for the sole purpose of executing web method calls) from accessing the user interface.

Usage

Used to update one or more existing active or retired users using an XML formatted file. The XML user edit is a batch process that updates multiple user records at a time. The success or failure message of each record follows the same method as the bulk import which can be emailed as well as logged.

Parameters

Parameter	Required	Description	Type
strSecurityToken	✓	Valid Session ID for consuming service	simple string
xmlUserData	✓	XML record formatted as described in Chapter 2 of the File Import Guide	simple string

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

String containing XML record with error description

Code examples

To update an two active Onboarding users from an XML record.

```
string sUsers = "<?xml version='1.0'?>"+ "<users>"+
 "<user><UserToEdit_AuthParam>Jane.Brown05</UserToEdit_AuthParam>"+
 "<TeamToAdd>Boston</TeamToAdd></user>"+
 "<user><UserToEdit_AuthParam>John.Smith</UserToEdit_AuthParam>"+
 "<TeamToAdd>Boston</TeamToAdd></user></users>"


Eprise.EpsStringEx ret = service.XMLUserEdit(sessionnum,sUsers); if (ret.ErrorNum != 1)
{
 Console.Out.WriteLine("Error while trying import users" + " Error:" +
ret.ErrorString);
 return;
}
```

Example return ret.Data:

```
<?xml version="1.0"?>
 <BulkImportResults>
 <Note>2 users modified</Note>
</BulkImportResults>
```

User requirements to execute XMLEdit Post

A user executing the XML Edit script must be a member of a team with "EditAllUsers" privileges

A user executing an XML Edit script that is retiring or unretiring users must be a member of a team with "Retire Users" and "Unretire Users" privileges.

Note: User is synonymous with employee.

XML Edit record

Example XML Edit record

The XML edit record must include:

- a unique criteria to identify a single user to update

- Six employee type identifiers are available to choose from: Guid, LoginID, SSOAuthParam, Email, and Employee_HRISID.
- The historical syntax for four of the unique identifiers is available for backward compatibility. The naming convention of elements used for unique identifiers (Guid, LoginID, AuthParam, Email) are prefixed with "UserToEdit_".

- the attribute(s) to be updated

The unique criteria can be the authentication parameter (for external authentication users), the user GUID, the login ID (for standard users), or the user email address. Note that the user email address must be a valid email address that uniquely identifies the user. Onboarding *does* allow both blank and non-unique email address for new users during user creation.

The record format is:

- The root element is <users>
- The parent element, <user> occurs once for each user to be modified
- The subchild elements reflect the fields in the user form, team additions, team removals, and a deletion flag

The following example shows editing two employees:

- Adding a user to a new team called Supervisors and removing the employee from the Engineering department (department internal code is ENG105)
- Add the employee to the "Assignees" team as a member and a controller, remove the employee from the "Bulk Importers" team as a member and controller. This employee is also given new category assignments for Location and Department. Note that existing category assignments are removed before new assignments are added.

```
<users>
 <user>
   <UserToEdit_AuthParam>johndoe@somecompany.com</UserToEdit_AuthParam>
   <TeamToAdd>Supervisors</TeamToAdd>
   <RemoveCategory>
    <Name>Department</Name>
    <Value>ENG105</Value>
    <Type>Code</Type>
   </RemoveCategory>
 </user>
 <user>
  <UserToEdit_AuthParam>janicesmith@somecompany.com</UserToEdit_AuthParam>
  <TeamToAdd>Assignees</TeamToAdd>
  <AddTeamController>Assignees</AddTeamController>
  <TeamToRemove>Bulk Importers</TeamToRemove>
  <RemoveTeamController>Bulk Importers</RemoveTeamController>
  <RemoveAllCategories/>
  <AddCategory>
   <Name>Location</Name>
   <Value>NE_BOSTON</Value>
   <Type>Code</Type>
  </AddCategory>
  <AddCategory>
   <Name>Department</Name>
   <Value>ENG105</Value>
   <Type>Code</Type>
  </AddCategory>
 </user>
```

```
29  </users>
```

Edit record required fields

Each record must contain *one* of the following unique identifiers. As noted in the sample XML record, only one of the identifiers should be included. You specify on EmployeeIDType and then the identifier value in the ID node.

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	✓	XML Document header	
<Users>	✓	Parent node	
<User>	✓	Parent node	One for each user
<UserToEdit_EmployeeID>	✓	Parent node for Filter	
<EmployeeIDType></EmployeeIDType>	✓	ID type	Possible values for EmployeeIDType are: • Employee_HRISID • LoginID • SSOAuthParam • Email • Guid • LifeSuiteID example: <EmployeeIDType>LoginID</EmployeeIDType>
<ID></ID>	✓	ID value	Specific to unique identifier for each user. This value is based on the Employee ID Type Example: <ID>Tammy.Jones</ID>
</UserToEdit_EmployeeID>	✓	close node	
</User>	✓	close parent node	Required following the edit fields listed below.

Select Users

```
1  <users>
2   <user>
3    <UserToEdit_EmployeeID>
4      <EmployeeIDType>Email</EmployeeIDType>
```

```
      <ID>James.Jones@SilkRoad.com</ID>
    </UserToEdit_EmployeeID>
    <First_Name>Jimmy</First_Name>
  </user>
  <user>
    <UserToEdit_EmployeeID>
      <EmployeeIDType>LoginID</EmployeeIDType>
      <ID>James.Jones</ID>
    </UserToEdit_EmployeeID>
    <First_Name>Jimmy</First_Name>
  </user>
  <user>
    <UserToEdit_EmployeeID>
      <EmployeeIDType>SSOAuthParam</EmployeeIDType>
      <ID>JJones@SilkRoad.com</ID>
    </UserToEdit_EmployeeID>
    <First_Name>Jimmy</First_Name>
  </user>
  <user>
    <UserToEdit_EmployeeID>
      <EmployeeIDType>Employee_HRISID</EmployeeIDType>
      <ID>20051148</ID>
    </UserToEdit_EmployeeID>
    <First_Name>Jimmy</First_Name>
  </user>
  <user>
    <UserToEdit_EmployeeID>
      <EmployeeIDType>Guid</EmployeeIDType>
      <ID>9b5f56b1-76d4-4c88-9222-070c13d922f3</ID>
    </UserToEdit_EmployeeID>
    <First_Name>Jimmy</First_Name>
  </user>
  <user>
    <UserToEdit_EmployeeID>
      <EmployeeIDType>LifeSuiteID</EmployeeIDType>
      <ID>7a231379-974c-4fc2-98cd-7580287b327d</ID>
    </UserToEdit_EmployeeID>
    <First_Name>Jimmy</First_Name>
  </user>
</users>
```

An alternate syntax is also available for backward compatibility reasons:

PrimaryIdentifier	Description
UserToEdit_Guid	Onboarding creates and stores a GUID for each user. You

PrimaryIdentifier	Description
UserToEdit_LoginID	Onboarding LoginID for users using Standard Authenticationwould not have this value unless it was programaticly provided.
UserToEdit_AuthParam	LoginID for users using External Authentication
UserToEdit_Email	Email address if the user email address is **unique**. Note: If the user email is not unique, this is not a valid unique criteria to use as a primary identifier.

Legacy User Select

```
<users>
 <user>
  <UserToEdit_Email>James.Jones@SilkRoad.com</UserToEdit_Email>
  <First_Name>Jimmy</First_Name>
 </user>
 <user>
  <UserToEdit_AuthParam>JJones@SilkRoad.com</UserToEdit_AuthParam>
  <First_Name>Jimmy</First_Name>
 </user>
 <user>
  <UserToEdit_LoginID>James.Jones</UserToEdit_LoginID>
  <First_Name>Jimmy</First_Name>
 </user>
 <user>
  <UserToEdit_Guid>9b5f56b1-76d4-4c88-9222-070c13d922f3</UserToEdit_Guid>
  <First_Name>Jimmy</First_Name>
 </user>
</users>
```

Edit record field descriptions

Each of sub-child elements of the <user> record are described below.

All <user> records require a primary key (see above) and the attribute or attributes to be updated.

The following elements are available for update the attributes of active employees such as employee profile information, team membership, team controller, assignment categories. You can also retire an employee through the edit record.

Note: Event information is *not* available in the XML **edit** record. See information on the UpdateEvents method.

Employee Profile Syntax

If an employee is retired and subsequently unretired, the "Employee Profile" fields will be persist through the retire process.

Element Syntax	Description of Expected Value
LoginID	User name used to login using 'standard' (Onboarding) authentication.
Employee_HRISID	Client controlled unique identifier.
AuthParam OR SSOAuthParam	SSOAuthParam is a unique userid. AuthParam syntax equates to SSOAuthParam. If the user is authenticated using an external application, the SSOAuthParam is the external application userid. The SSOAuthParam is used to log in to Onboarding for External Authentication. If an employee record is set to "Standard" authentication type, but your company has another system establishing a unique identifier, SilkRoad recommends you populate the SSOAuthParam with the alternate identifier. This facilitates maintaining employee data across multiple systems disregarding the Onboarding authentication method.
AuthType	Valid values for authentication type are: Standard - Onboarding authentication (LoginID and password are stored in Onboarding) CUProfile - Cached User Profile if an external application is used to authenticate the user. If <AuthType>CUProfile</AuthType> is set, the <SSOAuthParam> value is used as the login ID for Onboarding. Password is not stored in Onboarding.
Email	Employee email address.
First_Name	Employee first name.
Middle_Name	Employee middle name.
Last_Name	Employee last name.
MobileCountryCode	Must have SMS enabled to update field. Required if profile is being opted in to receive text messages. Must provide the 1,2 or 3 digit Country Code Codes: https://countrycode.org/ Example element syntax: <MobileCountryCode>1</MobileCountryCode>

Element Syntax	Description of Expected Value
PhoneMobile	When SMS is enabled the Mobile number must be added without Country Code
TimeZone	Must have SMS enabled to update field. Required if profile is being opted in to receive text messages. Valid values are: Allowed Time Zones Example Syntax:<TimeZone>1</TimeZone>
SmsOptIn	Must have SMS enabled to update field. Valid values are: 1 - Employee will be opted into receiving text messages 0 - Employee will be opted out of receiving text messages Example Syntax:<TimeZone>Eastern Standard Time</TimeZone>
ForcePasswordChange	Option to require an employee to change their password the next time they log in. They will be prompted for their original password, a new password and a confirmation of the new password. Valid values are: 1 - Employee will be forced to modify password at next login 0 - (default) Employee will be not be prompted to modify password at next log in Example Syntax:<ForcePasswordChange>1</ForcePasswordChange>
HireDate	Optional for active user. Expected format is YYYY-MM-DD. HireDate is recommended for all active employees and is referenced by rules that apply to your company's legal obligation to retain I-9 forms on file. If an I-9 Form exists for an employee, the employee profile Hire Date must match the I-9 form "begin employment date" in section 2 of the I-9 form. If the <HireDate> is updated, the begin employment date on Section 2 of the I-9 form will also be updated. Example Syntax: <HireDate>2009-09-15</HireDate>

Hire Date Updates

Employee Profile Hire Date and the I-9 Form Begin Employment Date

If an employee has an I-9 form, the Hire Date field on the Employee Profile must match the employment begin date in Section 2 of the I-9 form.

An update to the employee profile via the <HireDate> field, will automatically update the employment begin date field in Section 2 of the I-9 form.The XMLEdit rules follow the same behavoir as the Onboarding user interface.

The rules are as follows:

1. If an employee is added to Onboarding and the hire date is left blank the hire date can be populated by setting the hire date value in the employee profile.
2. If any I-9 form task is open, the employee begin date in section 2 can be updated by a privileged employee.
3. If the I-9 Form tasks are completed, and the hire date changes, the employer is not required to
    - attest to Section 2 of the I-9 form again
      OR
    - re-submit the employee to E-Verify.
4. Onboarding will automatically update the I-9 form with an accurate hire date and initial the change on the I-9 form. If the update is done through the Onboarding user interface, the employer adds their initials. If the update is done through XMLEdit, Onboarding uses the initials of the user specificied in the "I-9 Administrator for bulk upload modifications" configuration setting.

The "I-9 Administrator for bulk upload modifications" configuration setting is selected through the Onboarding user interface. A member of a team with "Manage Reverification Settings" privilege can select the employer (whose initials will be used) for a bulk edit modification. The configuration setting is selected through the menu option:

1. Choose Administration - I-9 Re-verification Settings menu selection from the home page.
2. For "I-9 Administrator for bulk upload modifications", choose the Edit link.
3. Choose Select to select from the filtered list of available employees. The available employees are filtered based on:
    a. employees with I-9 Administration privileges
    b. employees granted access to the Onboarding user interface.

E-Verify Fields

Element Syntax	Description of Expected Value
eVerifyCaseNumber	To edit an employee already processed through e-Verify you may include the DHS Case Number, Status, and Date for display on the I-9 Dashboard and Employee Profile - I-9 tab. DHS Case Number generated by an external system executing eVerify methods. eVerifyStatus must also be populated if eVerifyCaseNumber is populated. Example element syntax: <eVerifyCaseNumber>23435632</eVerifyCaseNumber>

Element Syntax	Description of Expected Value
eVerifyStatus	To edit an employee already processed through e-Verify you may include the DHS Case Number, Status, and Date for display on the I-9 Dashboard and Employee Profile - I-9 tab. Valid eVerify status codes are: IQ - Invalid Query SELFT - Self-Terminated EUAUTH - Resolved Unauthorized / Terminated SNTERM - Employee Not Terminated EAUTH - Resolved Authorizd eVerifyCaseNumber must also be populated if eVerifyStatus is populated. Example element syntax for an Authorized employee:<eVerifyStatus>EAUTH</eVerifyStatus>
eVerifyDate	To edit an employee already processed through e-Verify you may include the DHS Case Number you may import the DHS Case Number, Status, and Date for display on the Employee Profile - I-9 tab eVerify Case date. Expected format is YYYY-MM-DD. eVerifyDate will default to today's date if it is not inlcuded in the recored with eVerifyCaseNumber and eVerifyStatus Example: <eVerifyDate>2008-11-23</eVerifyDate>

Team Syntax

If an employee is retired the team membership is removed from the employee. If the employee is subsequently unretired, the Team fields can be used to add membership and controller team status.

Element Syntax	Description of Expected Value
TeamToAdd	A single team name to which the specified user will become a member. • this field can be specified one or many times within the <user> record. • The team name **can** contain spaces • The Team value is a match on the Team Name defined in the **Manage Teams** page.

Element Syntax	Description of Expected Value
AddTeamController	A single team name to which the specified user will become a controller. A team controller has the privilege to add or remove other members of a team. • this field can be specified one or many times within the <user> record. • The team name **can** contain spaces • The Team value is a match on the Team Name defined in the **Manage Teams** page • AddTeamController does not affect the team membership. Example:<AddTeamController>NE Coordinators</AddTeamController>
TeamToRemove	A single team name to remove the membership of the specified user • this field can be specified one or many times within the <user> record. • The team name rules match the TeamToAdd specifications • The specified user must be a member of the named team. Example:<TeamToRemove>NE Coordinators</TeamToRemove>
RemoveTeamController	A single team name to remove the controller privilege of the specified user • this field can be specified one or many times within the <user> record. • The team name rules match the TeamToAdd specifications • The specified user must be a controller of the named team. • RemoveTeamController does not affect the team membership. Example:<RemoveTeamController>NE Coordinators </RemoveTeamController>

Assignment Category Syntax

To update category assignments, the category syntax uses the same syntax to identify the category values as the BulkImport record. To update assignment categories the following criteria apply:

1. Before adding category assignments you must remove all existing assignments. In the <RemoveAllCategories/> tag in each record that includes the <AddCategory> tag.
2. <AddCategory> is used as the parent node instead of the <Category> syntax used in the Bulk Import record.
3. In addition to <Remove**All**Categories>, <RemoveCategory> is provided to remove a specific assignment category value from an employee. Use <RemoveCategory> to remove an assignment category without adding a new assignment category.
4. SilkRoad recommends using <Code> as the unique identifier for category assignment wherever possible. Other examples are included in the example code for syntax examples.
5. If an employee is retired the assignment categories are removed from the employee. If the employee is subsequently unretired, the assignment category fields can be used to re-add assignment categories.

Element Syntax	Description
RemoveAllCategories	Occurs once in the <User> record to remove all categories. This node is required before adding new assignment categories. Example: <RemoveAllCategories/>
RemoveCategory	Occurs for each assignment category remove from an employee. Parent node to Name/Value/Type category specification. Example syntax: `<RemoveCategory> <Name>Department</Name> <Value>D2_1</Value> <Type>Code</Type> </RemoveCategory>`
AddCategory	Occurs for each assignment category added to the employee. Parent node to Name/Value/Type category specification. Example syntax: `<AddCategory> <Name>Department</Name> <Value>D3_1</Value> <Type>Code</Type> </AddCategory>`
Name	Occurs once as a child element of <AddCategory> for each assignment category added or <RemoveCategory> for each assignment category removed. Valid values are text as it is entered in the Name field of the Manage Categories page. Example syntax: <Name>Location</Name>

Element Syntax	Description
Value	Occurs once as a child element of <AddCategory> for each assignment category added or <RemoveCategory> for each assignment category removed. Valid values are based on the <Type> element. If <Type>Name</Type> is specified, valid values are text as it is entered in the "Category Value" field of the Manage Categories page. Example syntax: <Value>NE_BOS</Value> <Type>Code/Type> is the recommended Name/Value/Type set to use. <Type>Name</Type> can not be used if duplicate values exist within your category tree. See <Type>
Type	Occurs once as a child element of <AddCategory> for each assignment category added or <RemoveCategory> for each assignment category removed. Valid values are: • Code - value is the category code <Value>MA_BEDFORD</Value> <Type>Code</Type> • Name - value is the field name <Value>Bedford</Value> <Type>Name</Type> • Path - value contains path <Value>All Locations\United States\Massachusetts\Bedford</Value> <Type>Path</Type> <Type> determine what type of value is passed in the value field to associate the correct category with the event or user. <Type>Code</Type> is the recommended Name/Value/Type set to use. <Type>Name</Type> can **not** be used if duplicate values exist within your category tree. If a duplicate value is specified with <Type>Name</Type> the imported record will fail.

Key Property Syntax

To update key properties, the category, person, and date syntax uses the same syntax to identify the category values as the BulkImport record. To update an employee's key properties the following criteria apply:

1. Adding a value for a property that already exists will return an error.
2. Use the <SetKeyProperties> tag to set an existing key property value to an updated value.
3. To clear a key property value (remove the value) use <SetKeyProperties> tag as the parent node to set an existing key property value to a blank.

4. Setting value for a property that does not exist will be ignored.

Element Syntax	Description

SetKeyProperties	Occurs once in the <User> record to set key propoerty values. Parent node to categories, people, or date property specification.
	
	Example:
	
	`<SetKeyProperties>`
	` <Category>`
	`  <Name>Current Department</Name>`
	`  <Value>D2_1</Value>`
	`  <Type>Code</Type>`
	` </Category>`
	`</SetKeyProperties>`

A code example of clearing a key property is:

```
<?xml version="1.0"?>
<users>
 <user>
  <UserToEdit_EmployeeID>
   <EmployeeIDType>Email</EmployeeIDType>
   <ID>Jessica.Wallace@MyCompany.com</ID>
  </UserToEdit_EmployeeID>
  <SetKeyProperties>
   <Category>
    <Name>Current Location</Name>
    <Value></Value>
    <Type>Code</Type>
   </Category>
    <Person>
        <Name>Manager</Name>
        <EmployeeID>
            <ID>888</ID>
            <EmployeeIDType>Employee_HRISID</EmployeeIDType>
        </EmployeeID>
    </Person>
  </SetKeyProperties>
 </user>
</users>
```

User Purge Syntax

Purging an employee from Onboarding will delete all the information associated with the employee record. This includes:

- Employee Profile data
- Uploaded Documents
- Events and the assoiated tasks
- eForms includeding the Form I-9

Note: Form I-9 retention rules override the ability to purge employees from Onboarding.

A 'tombstone' of the employee record is maintained in Onboarding with the employee name, so the task that the employee completed and audited activities they completed are maintained. The employee name can be changed to meet business requirnements to scrub even the name from the system.

Since an employee's events and tasks are deleted when the employee is purged, any record of the activities completed on their behalf are removed from all the reports within SilkRoad Onboarding. For this reason purge should only be used when there information is no longer needed in Onboarding. The employee record can be edited to remove all private information, eForms and uploaded documents can be individually deleted and the record can be 'retired' to prevent the employee from using Onboarding.

Element Syntax	Description
PurgeUser	Parent node indicating that the employee being edited should be scheduled for purge.
PurgeDate	The purge date cannot be the current date.
PurgeReason	Valid values are: • 0 : TerminatedEmployee - designed to be used when an employee no longer works for the organization and their data must be removed from Onboarding. • 1 : ExternalSystem - designed to be used when all the employee data has been moved from Onboarding to a system of record and the employee no longer needs to be in Onboarding. • 2 : GDPR - designed to be used for the GDPR 'right to be forgotten' and remove the employee information from Onboarding. • 3: Test Employee • 4: Duplicate or Erroneous Account

Example to purge user

```
<?xml version="1.0"?>
<users>
 <user>
  <UserToEdit_EmployeeID>
   <EmployeeIDType>Email</EmployeeIDType>
```

```
    <ID>Jessica.Wallace@MyCompany.com</ID>
   </UserToEdit_EmployeeID>
   <PurgeUser>
    <PurgeDate>2018-05-25</PurgeDate>
    <PurgeReason>2</PurgeReason>
   </PurgeUser>
  </user>
 </users>
```

Terminate Event Syntax

Element Syntax	Description
TerminateEvents	Used to terminate all the active events that the employee being edited is the benefiter of. Not used with TerminateEvent, Name or Code.
TerminateEvent	Used to terminate an event for the emplyoee being editied. Must include either the Name OR Code node.
Name	Used to identify the event to be terminated by name (in the default lanaguage) Must be used with the TerminateEvent node. Example: <TerminateEvent> <Name>Onboarding</Name> </TerminateEvent>
Code	Used to identify the event to be terminated by Code. Must be used with the TerminateEvent node. Example: <TerminateEvent> <Code>Event_2</Code> </TerminateEvent>

Terminate all events

```
<?xml version="1.0"?>
<users>
```

```
 <user>
  <UserToEdit_EmployeeID>
   <EmployeeIDType>Email</EmployeeIDType>
   <ID>Jessica.Wallace@MyCompany.com</ID>
  </UserToEdit_EmployeeID>
  <TerminateEvents />
 </user>
</users>
```

Terminate Event by Name

```
<?xml version="1.0"?>
<users>
 <user>
  <UserToEdit_EmployeeID>
   <EmployeeIDType>Email</EmployeeIDType>
   <ID>Jessica.Wallace@MyCompany.com</ID>
  </UserToEdit_EmployeeID>
  <TerminateEvent>
   <Name>Onboarding</Name>
  </TerminateEvent>
 </user>
</users>
```

Terminate Event by Code

```
<?xml version="1.0"?>
<users>
 <user>
  <UserToEdit_EmployeeID>
   <EmployeeIDType>Email</EmployeeIDType>
   <ID>Jessica.Wallace@MyCompany.com</ID>
  </UserToEdit_EmployeeID>
  <TerminateEvent>
   <Code>Event_2</Code>
  </TerminateEvent>
 </user>
</users>
```

Note: Events can also be terminated when retiring a user (below).

Retire Syntax

Onboarding includes a "Retired" status for an employee terminated from your employment. Retired employees can no longer log in to Onboarding or any Onboarding managed portals.

1. Retired employees require a termination date, if none is provided in the <RetireUser> record, and one does not already exist on the employee record, the current date will be used as a default.
2. If a record attempts to retire an employee who already exists with a retired status, a warning will be returned.
3. If a <RetireUser/> child node is included in a <User> record, the <TerminationDate> will be applied if it is included. All other edit tags (child nodes) included in that user record will be ignored.

Retire syntax is as follows:

Element Syntax	Description of Expected Value
RetireUser	Child node with no value. If included in a <User> record, the employee will be retired. Retired employees can not log in to Onboarding or Onboarding portals. Example: <RetireUser/>
TerminationDate	Employee termination date. Employees who are retired from Onboarding, require a termination date. Expected format is YYYY-MM-DD If the <RetireUser/> node exists in the same transaction, if the <TerminationDate> is not provided and does not already exist on the employee record the TerminationDate will default to today's date. Example: <TerminationDate>2008-11-23</TerminationDate> ⓘ Note: Termination date can be set outside of the <RetireUser> syntax

TerminateEvents	Child node of the <RetireUser> tag with no value.
	Employees who are retired, can not have any ongoing events. If included in a <User> record, the all of the employee events will be terminated (synonymous to cancelled).
	Example:
	`<RetireUser>`
	`  <TerminateEvents/>`
	`</RetireUser>`

Element Syntax	Description of Expected Value

IgnoreErrors	Child node of the <RetireUser> tag with no value.
	
	In order for an Employee to be retired all their assignments should be reassigned, if that is not possible when making updates through the API, a user being retired can have the assignment errors ignored. If included in a <User> record, the errors that would prevent the employee from being retired will be ignored, this could result in tasks being assigned to a retired user. The assignments tab of employees retired in this way should be reviewed to ensure tasks are completed.
	
	`<RetireUser>`
	`  <IgnoreErrors/>`
	`</RetireUser>`

Example syntax of editing three users:

1. retiring a user with ongoing events
2. retiring a user with a termination date
3. retiring a user ignoring and errors

```
<users>
 <user>
  <UserToEdit_EmployeeID>
   <EmployeeIDType>SSOAuthParam</EmployeeIDType>
   <ID>HHoward</ID>
  </UserToEdit_EmployeeID>
  <RetireUser>
   <TerminateEvents/>
  </RetireUser>
 </user>
 <user>
  <UserToEdit_AuthParam>RHoward</UserToEdit_AuthParm>
  <RetireUser>
   <TerminationDate>2008-11-24</TerminationDate>
  </RetireUser>
 </user>
 <user>
  <UserToEdit_AuthParam>OHoward</UserToEdit_AuthParm>
  <RetireUser>
   <IgnoreErrors/>
  </RetireUser>
 </user>
</users>
```

Unretire Syntax

A retired employee can be re-activated through executing the <Unretire/> command. Unretired employee are made active with their previous employee profile settings. Team membership and assignment category values must be reapplied to an unretired user.

- If an <UnretireUser/> child node is included in a <User> record, any other edit tags (child nodes) included in that user record will also applied. For example, an employee can be unretired, added to a team, and have assignment category values applied in the same transaction.
- If unretired employee's I-9 documents have been purged, the employee will be listed on purged reports. If the user is retired again, their I-9 documents may be purged again subsequent to that. Thus, an employee who has been retired more than once may have I-9 documents purged more than once.

Unretire syntax is as follows:

Element Syntax	Description of Expected Value
UnretireUser	Child node with no value. If included in a <User> record, the employee will be reset to active status. Example:<UnretireUser/>

Example syntax:

- unretiring a user and adding them to a team in the same transaction

```
<users>
 <user>
  <UserToEdit_EmployeeID>
   <EmployeeIDType>SSOAuthParam</EmployeeIDType>
   <ID>AWilson</ID>
  </UserToEdit_EmployeeID>
  <UnretireUser/>
  <TeamToAdd>NE_Coordinators</TeamToAdd>
 </user>
</users>
```

Change LoginID Syntax

The LoginID of a user can be changed by a user with the Change User LoginID privilege using the edit user API.

- When changing the LoginID, both <LoginID> and <LoginIDConfirm> nodes must be present, with the same value.
- The user accessing the API must have the Change User LoginID privilege.
- Changing the Login ID can be done along with updates to other fields.
- Once used LoginIDs are reserved and cannot be used again.

Change LoginID syntax is as follows:

Element Syntax	Description of Expected Value
LoginID	The new value of the LoginID for the user being edited
LoginIDConfirm	Confirmation of the LoginID, to ensure the LoginID change is intentional.

Example syntax:

- Changing the LoginID of a user. (This can be done along with updates to other fields)

```
<users>
 <user>
  <UserToEdit_EmployeeID>
   <EmployeeIDType>SSOAuthParam</EmployeeIDType>
   <ID>AWilson</ID>
  </UserToEdit_EmployeeID>
  <LoginID>AWilsonNewID</LoginID>
  <LoginIDConfirm>AWilsonNewID</LoginIDConfirm>
 </user>
</users>
```

Shared Dashboard Syntax

Editing an existing employee to allow them to access another employee's dashboard through the shared dashboard is available.

> Note: For more information on Shared Dashboards see Onboarding online help.

Similar to the bulk import syntax for events, the specific syntax used to create the category criteria used to filter a shared dashboard are configured based on your company's implementation of Onboarding. The XML syntax is based on what you have defined in your implementation.

You should cross reference the Onboarding Manage Shared Dashboard pages to understand the valid values of the defined categories.

> Note: If the <SharedDashboards> node is included in the employee edit syntax, the user executing the script must be a member of a team assigned the "Manage Shared Dashboards" privilege.

The syntax rules for SharedDashboards are:

1. Within the <SharedDashboard**s**> parent node, each Shared Dashboard is defined as a child node

<SharedDashboard>
</SharedDashboard>

1. The employee(s) who will be able to view the shared dashboard of the edited employee are identified with *one* of the following unique identifiers. *One* of the following attributes of the SharedWithEmployee_ is required:
    - SharedWithEmployee_GUID
    - SharedWithEmployee_Email
    - SharedWithEmployee_AuthParam

- SharedWithEmployee_LoginID

2. The SharedWithEmployee reference must be unique to the employee being edited.

If you want to change the criteria of an existing Shared Dashboard (with a unique employee) you must remove the shared employee first.

- <RemoveSharedDashboard> to remove a single employee share
- </RemoveAll> to remove all the employees sharing a dashboard

are available to remove a specific share or all shares before creating new Shared Dashboards.

1. Three options are available regarding the activation of the Shared Dashboard:
   - immediate activation for a non specific date range, specify:

<Activated>True</Activated>

- for a date specific activation specify a start and end date in YYYY-MM-DD format:

<ActivationFromDate>2009-04-21</ActivationFromDate>
<ActivationToDate>2009-04-28</ActivationToDate>

- to establish a shared dashboard but not as currently activated, specify:

<Activated>False</Activated>

A Shared Dashboard can be delegated by one or more specific categories. The category syntax follow the category syntax used for assignment categories. Each Category is coupled by three child elements:

<Name> </Name>
<Value> </Value>
<Type> </Type>

1. An error will occur on the SharedDashboard node for the following reasons:
2. the SharedWithEmployee_* cannot be found,

categories or category values are invalid,

1. ActivationFromDate, ActivationToDate have invalid format, or do not form a valid range
2. Activated, ActivationFromDate, ActivationToDate conflict with each other

Listed below is example syntax of editing three shared dashboard associations for one employee ("HHoward" in this example):

1. add a new shared dashboard to view "Jenny.Jones'" dashboard for the specified activation period. Only tasks for employees in Bedford will be visible. All notifications will be sent to HHoward and Jenny.Jones
2. add a new shared inactive dashboard to view "Roger.Lee's" dashboard. The inactive dashboard must be activated to be viewed.
3. remove an existing shared dashboard (of "JKnowles") from the "Shared Dashboard" (listing) page. A "removed" dashboard reference can be active or inactive.

```
<users>
 <user>
  <UserToEdit_EmployeeID>
   <EmployeeIDType>SSOAuthParam</EmployeeIDType>
   <ID>HHoward</ID></UserToEdit_EmployeeID>
  </UserToEdit_EmployeeID>
  <SharedDashboards>
```

```
    <SharedDashboard>
     <SharedWithEmployee_LoginID>Jenny.Jones</SharedWithEmployee_LoginID>
     <ActivationFromDate>2009-04-21</ActivationFromDate>
     <ActivationToDate>2009-07-21</ActivationToDate>
     <Category>
      <Name>Location</Name>
      <Value>Bedford_MA</Value>
      <Type>Code</Type>
     </Category>
     <ForwardNotifications>All</ForwardNotifications>
    <SharedDashboard>
    <SharedDashboard>
     <SharedWithEmployee_Email>Roger.Lee@some.com</SharedWithEmployee_Email>
     <Activation>False</Activation>
    </SharedDashboard>
    <RemoveSharedDashboard>
     <SharedWithEmployee_AuthParam>JKnowles</SharedWithEmployee_AuthParam>
    </RemoveSharedDashboard>
   </SharedDashboards>
  </user>
 </users>
```

Shared Dashboards syntax is as follows:

Element Syntax	Required	Description
SharedDashboards	✓	<SharedDashboards> is a parent element. Occurs one time for each edited employee. Parent node of Shared Dashboard feature occurring once for each new employee Example: <SharedDashboards>
SharedDashboard	✓	Child node occurring once for each employee dashboard to be accessed. Example: <SharedDashboard>

Element Syntax	Required	Description
SharedWithEmployee_*LoginID*	✓	Unique identifier for the employee dashboard to be accessed. Optional identifiers are: • SharedWithEmployee_*LoginID* • SharedWithEmployee_*GUID* • SharedWithEmployee_*Email* • SharedWithEmployee_*AuthParam* *One* identifier is required for each <SharedDashboard> Examples: <SharedWithEmployee_LoginID>Jenny.Jones</SharedWithEmployee_LoginID> <SharedWithEmployee_GUID>000392e2-9a5c-4ceb-ab60b95a20e72c30</SharedWithEmployee_GUID> <SharedWithEmployee_Email>Jenny.Jones@some.com</SharedWithEmployee_Email> <SharedWithEmployee_AuthParam>JJones246</SharedWithEmployee_AuthParam>
Activation	⚠	Required only if an activation date range is not specified. Possible values: True - activate the shared dashboard immediately and indefinitely False - Shared Dashboard entry is created but not activated. It can be activated through the Share Dashboard page at a future date. Example syntax: <Activation>True</Activation>
ActivationFromDate		Shared Dashboard activation start date. Expected format is YYYY-MM-DD. From date must be before the To date. Example syntax: <ActivationFromDate>2009-04-21</ActivationFromDate>
ActivationToDate		Shared Dashboard activation end date. Expected format is YYYY-MM-DD. To date must be after the From date. Example syntax: <ActivationToDate>2009-04-28</ActivationToDate>

Element Syntax	Required	Description
Category		Occurs for each category defined on the event Example syntax: <Category> <Name>Location</Name> <Value>MA_BEDFORD</Value> <Type>Code</Type> </Category>
Name		See Category Syntax (see page 26) for additional information
Value		See Category Syntax (see page 26) for additional information
Type		See Category Syntax (see page 26) for additional information
ForwardNotifications		Possible values are: All - All notifications related to the shared dashboard tasks will be forwarded to the alternate (new employee) None- None of the notifications related to the shared dashboard tasks will be forwarded to the alternate (new employee) WarningsOverdue - Notifications related only Warnings or Overdue notices related to the shared dashboard tasks will be forwarded to the alternate (new employee) Default is All Example syntax: <ForwardNotifications>All</ForwardNotifications>

XMLUserEdit Errors

Each XML User Edit error is communicated with the following element:

```
<UserError>
 <Error>
  <Type></Type>
  <Message></Message>
  <ErrorCode></ErrorCode>
 </Error>
</UserError>
```

If an error exists, the output includes a <UserError> tag. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<BulkEditResults>
 <Note>0 users were modified</Note>
 <Note>1 user skipped because they had errors</Note>
 <UserError>
  <Name>The 1st user node (Login ID: Shane.Veen)</Name>
  <IdentifierNode>
   <EmployeeIDType>Email</EmployeeIDType>
   <ID>sveen@mysrt.com</ID>
  </IdentifierNode>
  <Error>
   <Type>Invalid Input</Type>
   <Message>Invalid date value entered for HireDate date, must be in YYYY- MM-DD format: 01-19-2012</Message>
   <Code>30037</Code>
  </Error>
 </UserError>
 <BadUsers>
  <user>
   <UserToEdit_EmployeeID>
    <EmployeeIDType>Email</EmployeeIDType>
    <ID>sveen@mysrt.com</ID>
   </UserToEdit_EmployeeID>
   <HireDate>01-19-2012</HireDate>
  </user>
 </BadUsers>
</BulkEditResults>
```

XMLUserEdit Error Messages

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The SSOAuthParam value is already in use by another user	21022	Attempt to modify the AuthParam or SSOAuthParam to a value used by an existing user NOTE: this errorcode value was 0 prior to 2.8.0. You must disassociate the default XMLUserEdit transformation to return this errorcode

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The Employee_HRISID is already in use by another user	20125	Attempt to modify the Employee_HRISID to a value used by an existing user NOTE: this errorcode value was 0 prior to 2.8.0. You must disassociate the default XMLUserEdit transformation to return this errorcode
Invalid Input	The LifeSuiteID value is already in use by another user	20126	RESERVED For SilkRoad Attempt to modify the LifeSuiteID to a value used by an existing user NOTE: this errorcode value was 0 prior to 2.8.0. You must disassociate the default XMLUserEdit transformation to return this errorcode
Fatal Error	Parsing failed due to: *message*	30001	Caused by malformed XML
Invalid Input	The e-mail address used to identify a user is used by more than one user	30013	
Invalid Input	"{0} event doesn't exist for user: {1}"	30014	0 - Name of the event in the default lanauge 1 - Name of the user
Invalid Input	Missing category name node under category node	30030	Each <Category> occurence must have a <Name> node
Invalid Input	Invalid category value type (Type entered: *x*) must be 'Name', 'Code', or 'Path'	30031	Valid <Type> values are: <Type>Name</Type> <Type>Code</Type> <Type>Path</Type>
Invalid Input	Invalid event name '*x*' found	30032	Undefined <Event><Name> value was specified.
Invalid Input	Parsing failed due to: *x*	30033	Unrecognized node specified.

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Unknown date name: *x*	30034	Undefined <Date> value specified
Invalid Input	Invalid manager identifier entered for *x* manager: {1}	30035	Undefined manager name specified on event
Invalid Input	Invalid manager name entered: {0}");	30036	Invalid <Value> for the coorsponding <Person> <Name>
Invalid Input	Invalid date value entered for *x* date, must be in YYYY-MM-DD format: {1}	30037	
Invalid Input	User marked as ForceActiveNewHire, but does not have required *x data*	30038	Required fields must be included in the ForceActiveNewHire record.
Invalid Input	Termination Date not in YYYY-MM-DD format: *date*	31001	Invalid date format
Invalid Input	The e-mail address (address) which was specified for the *nith* user node is not used by any user in the system	30012	User identified by email address can not be found
Invalid Input	The e-mail address (address) which was specified for the *nith* user node is used by n users so it cannot be used to uniquely identify a single user	30013	email address is not a unique identifier in your Onboarding installation.
Invalid Input	{0} event doesn't exist for user: {1}	30014	0 - Event Name 1 - Employee name
Invalid Input	More than one category value name found for value category: *category name*	30024	<Type>Name</Type> was specified but Name is not a unique category identifier
Invalid Input	Missing eVerify status data	31011	<eVerifyCaseNumber> and <eVerifyStatus> must both have values if either have values.
Invalid Input	Missing eVerify case number	31012	<eVerifyCaseNumber> and <eVerifyStatus> must both have values if either have values.

<Type>	<Message>	<ErrorCode>	Description
Fatal Error	XML user edit failed due to: *message*	31020	
Unknown User	No user could be found for the *ith* user node	31021	<User> node found with no child nodes
User Validation	No user identification node could be found for the *ith* user node	31022	Invalid UserToEdit_GUID was specified.
Invalid Input	You cannot retire a retired employee	31023	
Invalid Input	You cannot unretire an employee who is not retired	31024	
Invalid Input	Employee could not be removed as a controller from the named team because that employee is not a controller of the team	31025	
Invalid Input	Employee could not be removed as a member from the named team because that employee is not a member of the team	31026	
Invalid Input	User could not be added to the named team because you are not a controller for that team	31027	
Invalid Input	User could not be removed from the named team because you are not a controller for that team	31028	
Invalid Input	The team named *name* is unknown	31029	
Invalid Input	Cannot add employee to assignment category because the employee is already assigned to that category	31030	
Invalid Input	Cannot remove employee from assignment category because the employee is not assigned to that category	31031	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The named field is not editable or unknown. Because the XmlEditStopOnNonEditableField setting is enabled this user will be skipped.	31032	
Invalid Input	The named field is not editable or unknown.	31033	
User Save Error	Failed to save user: *message*	31034	
Invalid Input	The user invoking the bulk import does not have the privilege to create active users.	31035	Employee who obtained the SessionID must be on a team with the “Create User” privilege.
Invalid Input	XML user edit is stopping because an error was found and XMLUploadStopOnError is set to 1	31036	XMLUploadStopOnError set through the Administration ->Settings page. Set XMLUploadStopOnError to False to allow user edit to continue modifying valid records.
Invalid Input	The e-mail address (x) which was specified for the n user node is not used by any user in the system	31037	Invalid email address specified.
Invalid Input	The e-mail address (x) which was specified for the {n} user node is used by {nn} users so it cannot be used to uniquely identify a single user	31038	Non unique email address specified.
Invalid Input	An unkown error has occured	31039	
Invalid Input	Failed to load XML document (x)	31040	
Invalid Input	XML user edit failed due to:{x}	31041	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	No user could be found for the {*x*} user node	31042	For example: if an invalid LoginID, Employee_HRISID (no old syntax), SSOAuthParam (old syntax - AuthParam), Guid, LifeSuiteID the return will be: <Error> <Type>Invalid Input</Type> <Message>No user could be found for the 1st user node</Message> <Code>31042</Code> </Error>
Invalid Input	Unkown abstract category: {*x*}	31043	Unknow "Alias Name" category reference specified.
Invalid Input	The UserToEdit_*x* identifier node for the *y* user cannot be blank	31044	Employee identifier included in *y* (number) the record is blank.
Invalid Input	Can't retire user:	31051	The employee being retired is a named person (employers) referenced on active events or not completed tasks. You must reassign the employer's task and/or events and try to retire again.
Invalid Input	Purge Date must be at least 24 hours in the future.	31063	
Invalid Input	Purge Date not in YYYY-MM-DD format:	31062	
Invalid Input	Purge Reason must be a number. Valid values are: 0:TerminatedEmployee,1:ExternalSystem,2:GDPR,	31064	
Invalid Input	Missing required information: Terminate Event: You must specify an Event Name or Code as the first sub-node of the TerminateEvent node.	3400	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Insufficient Privileges to perform the action	34001	User executing script does not have the correct privilege (team membership or event coordinator designation) to execute the referenced action
Invalid Input	The I-9 Administrator for bulk upload modifications setting is not set. Updating a user's Hire Date on an I- 9 through bulk upload requires this setting	35011	If an employee has an I-9 form a <HireDate> update affects the I-9 form section 2 begin employment date. A required configuration settting has not been set. See "Hire Date Updates" for additional information.
Invalid Input	Key Property: Invalid Category Name - 'LOCATION'	39001	
Invalid Input	Key Property: Invalid Category Code - 'CATEGORY_1'	39002	
Invalid Input	Key Property: Invalid Person Name - 'MANAGER'	39003	
Invalid Input	Key Property: Invalid Person Code - 'PERSON_1'	39004	
Invalid Input	Key Property: Invalid Date Name - 'START	39005	
Invalid Input	Key Property: Invalid Date Code - 'DATE_1'	39006	
Invalid Input	The following key properties are required but do not have a value: 'LIST OF MISSING REQUIRED KEY PROPERTIES'	39007	
Invalid Input	Key Property: Invalid category value name for 'LOCATION' category: 'FAKE TOWN'	39008	
Invalid Input	Key Property: Invalid category value path for ''LOCATION category: 'FAKE/ PATH/4U'	39009	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Key Property: Invalid category value code for 'LOCATION' category: 'FAKE_CODE'	39010	
Invalid Input	Key Property: Invalid manager identifier entered for 'MANAGER' manager: 'FAKE.LOGIN'	39011	
Invalid Input	Key Property: Date format Key Property: Date format	39012	Expected format is YYYY-MM-DD

Retrieve User Data API

Overview

Retrieve User attributes

The following methods are available to retrieve employee attributes:

- **GetUserIDList** accepts a synchronization token returning a list of userIDs and returns an XML document containing one or more users meeting the synchronization criteria. The user IDs can be used as input to the GetUserProfile, GetUserProfileEx, GetUserProfileEx2 methods to retrieve specific user attributes. GetUserIDList allows you to systematically retrieve all the users in the system and subsequently retrieve all the users that have been modified since the successfully processed user.
- **GetUserProfileEx** and **GetUserProfileEx2** return all attributes associated with any active Onboarding user. GetUserProfileEx2 includes additional fields including additional E-Verify case data, and two additional user id identifiers. Both methods output an xml document with name\value pairs for all attributes.
- **GetUserProfile** returns an array of attributes associated with any active Onboarding user. GetUserProfile returns a subset of the attributes returned by GetUserProfileEx. The result of GetUserProfile provides a table with two arrays of data. The first array is a list of standard and extended user attribute names. The second array contains the corresponding values.

GetUserIDList

Usage

GetUserIDList returns a list of user IDs that match search criteria. The list can be all the users in the system or all the users who have been updated since the specified synchronization token marker. The synchronization token marker is a system value and not a user-generated one. For subsequent calls, you will populate the <synchronizationToken> from the result of GetUserIDList to return the user updates since the successfully processed user record. The users are returned sorted by last modified date time. The last user in the list is the most recent modified user in the system.

To return all the users in Onboarding, call GetUserIDList passing a null for the synchronization token. Onboarding returns user IDs listing every user in the system ordered by last modified.

To retrieve a set of users that have been updated since a point in time, call GetUserIDList using the last synchronization token return in the previous results set.

By default, your GetUserIDList results set is limited to 1000 employees per call. The maximum results set is defaulted to 1000 employees. If you want to change the maximum results set size, contact SilkRoad Support. For performance reasons, SilkRoad does not recommend modifying the results set size to more than 1000 rows, but it is configurable if your integration justifies a different (smaller) paging value. The intent of the maximum is for the calling app to page through the results set.

For example, assume the maximum records configuration (MaxRecords) is set to 1000. If you are calling GeUserIDList and your results set is < MaxRecords then you have returned all the records updated since the last synchronization token. If the number of rows returned equals the MaxRecords, call the method again (passing in the last synchronization token value) and return the next 1000 records. Continue calling GetUserIDList until the results set is less than MaxRecords. A number less than the MaxRecords indicates the last page of data (no employee records have been updated since the last synchronization token).

When using this technique to synchronize data from Onboarding to an external system, it is important for your external system to allow duplicates.

Any updates to the following user attributes will be recognized as a user update:

- Employee details

- Employee User Profile
- Key properties
- Event status change (an event is launched, event completed, or event cancelled, including task updates that cause changes to the Event status)
- E-Verify Case data (Resolve Code, Resolve Date, Create Date, Referral Date,Close Date, Last Message,Manual Case Number,Manual Case Submitted By, Manual Resolve Date)

Assignment categories, team membership, modifying event values, task updates, and document uploads do not trigger a change to the user data.

For example after calling GetUserIDList with a synchronization token, the return list of IDs could be passed to GetUserProfileEx2 to obtain the employee information for each employee updated since the last call. If search criteria is specified that does not match any data, no data is returned.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below. The XML specifies your synchronization starting point. You may return all the users in the system (with their tokens) or return all the users modified since the token specified.	simple string

Examples of GetUserIDList strXML

Example of GetUserIDList strXML to return a list of all users and their current synchronization token in RedCarpet:

```
<?xml version="1.0" encoding="utf-8"?>
<GetUserIDListInput>
 <SynchronizationToken />
</GetUserIDListInput>
```

Example of GetUserIDList strXML to return a list of users updated since the specified token:

```
<?xml version="1.0" encoding="utf- 8"?>
<GetUserIDListInput>
 <SynchronizationToken>yx+c9VyG1HwjHsIo0j8z0
jd47aKGEJicwWywoYsnn7hm3wo+3KfH87DN3GNxFKWP0baBmbdcltuVKKwoFzDA7Q==</
SynchronizationToken>
</GetUserIDListInput>
```

XML Input Nodes

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	☐	XML Document header	
<GetUserIDListInput>	☐	Parent node	
<SynchronizationToken />	☐	Synchronization token marker for where to begin the results set. Null returns all user records in update order. The most in update order. The most recent update will be the last in the results set. A valid synchronization token (obtained by a previous call to this method) returns all users updated since the last call.	<SynchronizationToken /> indicates a null value. The synchronization token is a simple string and should be stored as a string.
</GetUserIDListInput>	☐	close parent node	

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

An xml document containing the user IDs of employees that meet the synchronization criteria. The nodes in the return data are as follows:

XML Nodes	Description
<?xml version="1.0" encoding="utf-8" ?>	XML document header
<GetUserIDListOutput xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">	Occurs once for each document
<Users>	Occurs once for each document
<User>	Occurs once for each user returned
<Guid></Guid>	Guid (see page 106) value returned for each user
<Employee_HRISID/>	Employee_HRISID (see page 0) value returned if populated. Optionally configured as required.
<SSOAuthParam />	SSOAuthParam (see page 0) value returned if populated. Configured if authentication type is external configuration. Optionally configured if standard authentication.
<LoginID></LoginID>	LoginID (see page 0) value is returned
<Email></Email>	Email (see page 0) value returned if populated.
<LifeSuiteID />	LifeSuiteID (see page 0) value returned if it has been populated
<EmployeeStatus>Active</ EmployeeStatus>	EmployeeStatus returns Active or Retired for each user.
<SynchronizationToken> SynchronizationToken>	Synchronization Token is a system generated value returned for each user. The token's sole purpose is to synchronize updates between RedCarpet and an external system.
</User>	End User node. Occurs once for each user

XML Nodes	Description
</Users>	End Users list. Occurs once for each document

GetUserIDList Error messages

<Type>	<Message>	<ErrorCode>	Description
Error	Invalid input XML: {0}	42000	
Error	The synchronization token was provided cannot be used for this web service method. It may have been returned from a different web service and cannot be used with this web service, with the criteria specified.	42001	Invalid Synchronization token
Error	The synchronization token cannot be decrypted and appears to have been tampered with	42002	Invalid Synchronization token

Examples

Code example

To return a list of all the users in RedCarpet:

```
d= service.GetUserIDList(sessionnum,
"<?xml version="1.0" encoding="utf-8"?><GetUserIDListInput><SynchronizationToken /
></GetUserIDListInput>")
```

Example output

This sample output includes only two users for readability.

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
 <![CDATA[
  <?xml version="1.0" encoding="utf-8"?>
   <GetUserIDListOutput xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
    <Users>
     <User>
      <Guid>56e6f1f7-aab1-43ca-9c5900c2b95d7087</Guid>
      <Employee_HRISID>1001</Employee_HRISID>
      <SSOAuthParam />
      <LoginID>Tammy.Jones</LoginID>
      <Email>tjones@MySRT.com</Email>
      <LifeSuiteID />
      <EmployeeStatus>Active</EmployeeStatus>

<SynchronizationToken>yx+c9VyG1HwjHsIo0j8z0jd47aKGEJicwWywoYsnn7hm3wo+3KfH87DN3GNxFKWPD
urca47G1jt1uqNOUx470Q==</SynchronizationToken>
     </User>
     <User>
      <Guid>0ce54d80-c93d-4844-842534a6863009c3</Guid>
      <Employee_HRISID />
      <SSOAuthParam />
      <LoginID>James.Jones</LoginID>
      <Email>jjone@MySRT.com</Email>
      <LifeSuiteID />
      <EmployeeStatus>Active</EmployeeStatus>
      <SynchronizationToken>yx+c9VyG1HwjHsIo0j8z0jd47aKGEJicwWywoYsnn7hm3wo+
3KfH87DN3GNxFKWPgTwFRsYrvF3u/PCOyIVOjQ==</SynchronizationToken>
     </User>
    </Users>
   </GetUserIDListOutput>
  ]]>
 </Data>
</EpsStringEx>
```

GetUserProfileEx2

Usage

Retrieves information on the employee including employee detail, employee profile fields, launched events and event data. Event data includes event named people, categories and dates, and e-verify case data if applicable.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

Example Structure of GetUserProfileEx2 strXML

```
<?xml version="1.0" encoding="utf-8"?>
<GetUserProfileEx2Input>
  <EmployeeIDFilter>
    <EmployeeIDType>LoginID</EmployeeIDType>
    <ID>Tammy.Jones</ID>
  </EmployeeIDFilter>
</GetUserProfileEx2Input>
```

Nodes

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	✓	XML Document header	
<GetUserProfileEx2Input>	✓	Parent node	
<EmployeeIDFilter>	✓	Parent node	
<EmployeeIDType></EmployeeIDType>	✓	ID type	Acceptable values are: • Employee_HRISID • LoginID • SSOAuthParam • Email • Guid • LifeSuiteID See "Employee ID Types" (see page 20) for descriptions. example:<EmployeeIDType>LoginID</EmployeeIDType>
<ID></ID>	✓	ID value	Specific to unique identifier for each user. This value is coupled to the Employee ID Type specified. Example: <ID>Tammy.Jones</ID>
</EmployeeIDFilter>	✓	close node	

Node	Required	Description	Notes
</ GetUserProfileEx2Input>	✓	close parent node	

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

An xml document with user data. User data includes: employee detail, employee profile fields, key properties, assignment categories, launched event (including event status) and the event data. Event data includes event named people, categories and dates, and e-verify case data if applicable.

Extended attributes are included in the string after the standard attributes.

GetUserProfileEx2 Error messages

<Type>	<Message>	<ErrorCode>	Description
UserError	The LoginId ({0}) used to identify a user is used by more than one user	31059	

<Type>	<Message>	<ErrorCode>	Description
UserError	No user could be found for the id: {0}	31060	

Examples

Example Output

```
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <GetUserProfileEx2Results>
    <users xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <user>
      <first_name>Tammy</first_name>
      <Middle_Initial xsi:nil="true" />
      <middle_name xsi:nil="true" />
      <last_name>Jones</last_name>
      <UserGUID>56e6f1f7-aab1-43ca-9c5900c2b95d7087</UserGUID>
      <LoginID>Tammy.Jones</LoginID>
      <AuthType>Standard</AuthType>
      <SSOAuthParam xsi:nil="true" />
      <Email><![CDATA[tjones@MySRT.com]]></Email>
      <Employee_HRISID>1001</Employee_HRISID>
      <LifeSuiteID xsi:nil="true" />
      <TerminationDate xsi:nil="true" />
      <HireDate>2012-12-04</HireDate>
      <Retired>0</Retired>
      <Full_Name xsi:nil="true" />
      <Preferred_Name xsi:nil="true" />
      <SSNO xsi:nil="true" />
      <DOB xsi:nil="true" />
      <SectionContactInfo xsi:nil="true" />
      <Address1 xsi:nil="true" />
      <Address2 xsi:nil="true" />
      <City>East Kingston</City>
      <State xsi:nil="true" />
      <Zip xsi:nil="true" />
      <categories>
       <Category>
        <CategoryCode>AbstractCategory_5</CategoryCode>
        <name>Location</name>
        <value>Boston</value>
        <Code>LOC_BOS</Code>
        <Path>\All Locations\Boston</Path>
```

```
        </Category>
       </categories>
       <teams>
        <Team isMember="true">Reporter</Team>
       </teams>
       <event>
        <name>Onboarding-EVerify</name>
        <EventID>7</EventID>
        <EventStatus>Complete</EventStatus>
        <EventCode>Event_7</EventCode>
        <Person>
         <Code>Person_1</Code>
         <name>Manager</name>
         <value>Robert Manager</value>
         <LoginID>Robert.Manager</LoginID>
         <SSOAuthParam />
         <Email>Robert.Manager@MySRT.com</Email>
         <Employee_HRISID>101</Employee_HRISID>
         <LifeSuiteID />
         <UserGUID>0251e061-1e67-42e2-b41469db2e6eb60a</UserGUID>
        </Person>
       </event>
       <EVerifyCases>
        <EVerifyCase>
         <CaseNumber>96607afd-293f-42be-b85d- fa66eeb109b9</CaseNumber>
         <ResolveCode>EELIG</ResolveCode>
         <ResolveDate>2012-12-04</ResolveDate>
         <CreateDate>2012-12-04</CreateDate>
         <ReferralDate xsi:nil="true" />
         <CloseDate>2012-12-04</CloseDate>
         <LastMessage>The employee continues to work for the employer after receiving an
Employment Authorized result.</ LastMessage>
         <ManualCaseNumber xsi:nil="true" />
         <ManualCaseSubmittedBy xsi:nil="true" />
         <ManualResolveDate xsi:nil="true" />
        </EVerifyCase>
       </EVerifyCases>
      </user>
     </users>
   </GetUserProfileEx2Results>
  </Data>
</EpsStringEx>
```

GetUserProfileEx

Usage

Retrieves standard and extended user attribute information for a specified user.

Parameters for GetUserProfileEx

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strLoginID	☐	Required if strFKUserID is not populated. Onboarding login ID on the employee profile. This is a unique identifier for the employee. This parameter is mutually exclusive of strFKUserID. Specify an empty string if you are specifying a value for strFKUserID.	simple string
strFKUserID	☐	Required if strLoginID is not populated. Alternate ID for an employee from another corporate application. This is a unique identifier for the employee. This parameter is mutually exclusive of strLogInID. Specify an empty string if you are specifying a value for strLogInID.	simple string
strAlternateUserID		Reserved for future use	simple string

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

An xml formatted string.

Extended attributes are included in the string after the standard attributes.

Examples

Code example

To return user attributes for a standard RedCarpet user with the external authentication parameter of "JJohnson":

```
Eprise.EpsTableEx t = service.GetUserProfileEx(sessionnum,"", "JJohnson","")
```

Example output

This sample output has the redundant nodes compressed for readability. This sample includes all the standard attributes and multiple extended attributes. The extended attributes examples include phone number, Interest drop-down list, and Days_avail comma separated multi-select.

This is an example of an employee benefiting from an event. This employee is not defined as a provider as evident by no assignment categories or team membership. Example category output is displayed in the event categories.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<users>
 <user>
 <First_Name>Jane</First_Name>
 <Middle_Initial>L</Middle_Initial>
 <Middle_Name>Lee</Middle_Name>
 <Last_Name>Johnson</Last_Name>
 <UserGUID>7c60d370-6679-40c1-a57411ddf41dcdb0</UserGUID>
 <LoginID>Jane.Johnson</LoginID>
 <AuthType>Standare</AuthType>
 <AuthParam>JJohnson<AuthParam/>
 <Email>jane.johnson@work.com</Email>
 <Phone />
 <Interests>Books</Interests>
 <Days_avail>1,3,5,7</Days_avail>
 <Categories />
 <Teams />
 <Event>
  <Name>Onboarding</Name>
  <EventID>1</EventID>
  <EventStatus>InProgress</EventStatus>
  <Person>
   <Name>Manager</Name>
   <Value>Trish Madison</Value>
   <LoginID>trish.madison</LoginID>
```

```
    <AuthParam />
    <Email>trish.madison@work.com</Email>
   </Person>
   <Person>
    <Name>HR Coordinator</Name>
    <Value>Harry Wise</Value>
    <LoginID>hwise</LoginID>
    <AuthParam />
    <Email>hwise@work.com</Email>
   </Person>
   <Category>
     <Name>Location</Name>
     <Value>Bedford</Value>
     <Code />
     <Path>\All Locations\United States\Massachusetts\Bedford</Path>
   </Category>
   <Category>
    <Name>Department</Name>
    <Value>Engineering</Value>
    <Code />
    <Path>\All Departments\Engineering</Path>
   </Category>
   <Category>
    <Name>Job Type</Name>
    <Value>Engineer</Value>
    <Code />
    <Path>\All Job Types\Engineer</Path>
   </Category>
   <Date>
     <Name>Start</Name>
     <Value>2008-07-25</Value>
   </Date>
  </Event>
 </user>
 </users>
```

GetUserProfile

Usage

Retrieves standard and extended user attribute information for a specified user. Method output is an array of strings.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string

Parameter	Required	Description	Type
strLoginID	☐	Required if strFKUserID is not populated. Onboarding login ID on the employee profile. This is a unique identifier for the employee. This parameter is mutually exclusive of strFKUserID. Specify an empty string if you are specifying a value for strFKUserID.	simple string
strFKUserID	☐	Required if strLoginID is not populated. Alternate ID for an employee from another corporate application. This is a unique identifier for the employee. This parameter is mutually exclusive of strLogInID. Specify an empty string if you are specifying a value for strLogInID.	simple string
strAlternateUserID		Reserved for future use	simple string

Returns

A table of type EpsTableEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsTableEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <ArrayOfString>
   <!-- Array of string 'column headers' -->
  </ArrayOfString>
  <ArrayOfString>
   <!-- One array of strings for each result record -->
  </ArrayOfString>
 </Data>
</EpsTableEx >
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Two string arrays. The first contains the attribute names. The second contains the corresponding values to the name. Extended attributes are included in the array after the standard attributes.

Examples

Code example

To return user attributes for a standard RedCarpet user with the login id of "john.james":

```
Eprise.EpsTableEx t = service.GetUserProfile(sessionnum,"john.james","")
```

Example output

This sample output includes all the standard attributes and one extended attribute called "EmergencyContactInfo".

```
<Data>
 <ArrayOfString>
<string>OBJECTID</string>
<string>USERGUID</string>
<string>LOGINID</string>
<string>FKUSERID</string>
<string>DEPARTMENT</string>
<string>LOCATION</string>
<string>POSITION</string>
<string>MANAGER_EMAIL</string>
<string>ONBOARDING_MANAGER_EMAIL</string>
<string>User_Type</string>
<string>User_ID</string>
<string>Hire_Date</string>
<string>First_Name</string>
<string>Middle_Initial</string>
<string>Last_Name</string>
<string>Email</string>
<string>PendingLoginId</string>
<string>EmergencyContactInfo</string>
</ArrayOfString>
</ArrayOfString>
<ArrayOfString>
<string>4478</string>
<string>e4cc00a5-79a4-4c47-8019e8fd005963f3</string>
<string>john.james</string>
<string>jjames<string/>
<string>Customer Service</string>
<string>East</string>
```

```
<string>Manager</string>
<string/>
<string/>
<string>Onboarding Manager</string>
<string>e4cc00a5-79a4-4c47-8019e8fd005963f3</string>
<string>2007-03-06</string>
<string>rc</string>
<string/>
<string>example</string>
<string><!\[CD[ATA\[example.user@silkroadtech.com|mailto:example.user@silkroadtech.com]
\]\]></string>
<string/>
<string><!\[CDATA\[Include name, phone, email address if applicable,\]\]></string>
</ArrayOfString>
</Data>
```

Forms API

Onboarding web methods provide the following options to access Onboarding forms:

- Retrieve saved forms and completed forms based on various filter criteria
- Retrieve PDF files (populated through Onboarding)
- Indicate if a form has already been retrieved
- Delete existing forms

Retrieve existing forms

Web methods are available to retrieve existing forms. Any method can be used to retrieve a list of Form ID's that meet specific criteria.

Options	GetFormsIDs	GetCompletedForms	GetFormsIDsEx	GetCompletedFormsEx	GetCompletedFormsList
Input	parameters	parameters	parameters	parameters	XML
Event Benefiter	✓	✓	✓	✓	✓
Form Type (template name)	✓	✓	✓	✓	✓
Delivered	✓	✓	✓	✓	✓
Completed (strVerified)	✓	assumed = true	✓	assumed = true	assumed = true
Date Range	form creation	form creation	form completion	form completion	both
Event		✓	✓	✓	✓
Event Categories			✓	✓	✓
Employee Key Properties					✓
Employee Status					✓
Event Status					✓

The result of the Forms methods provides a list (table) of unique Form IDs. For purposes of retrieving the actual form, each Form ID is passed to the GetFormXML (see page 137)or GetFormPDF (see page 139)methods to retrieve the form data or PDF.

Removed from PDF - To be removed from page after review.

GetCompletedFormsList

Usage

Retrieves a list of completed form IDs that meet the specified criteria.

If no forms meet this specified filter criteria, no form IDs are returned.

Input for GetCompletedFormsList

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

Nodes in GetCompletedFormsList

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	✓	XML Document header	
<GetCompletedFormsListInput>	✓	Parent node	
<FormFilter>		Parent Node	
<IsFormDelivered> </IsFormDelivered>		Optional Occurs once.	Expect values: "1" for delivered "0" for not delivered

Node	Required	Description	Notes
<CompletionDateRange> <BeginDate> </BeginDate> <EndDate> </EndDate> </CompletionDateRange>		Optional - date range of upload Occurs once.	If CompletionDateRange is specified both Begin Date and End Date must be specified. Fomat is YYYY-MM-DD
<FormOwnerFilter>		Parent Node	
<EmployeeIDFilter> <EmployeeIDType> </ EmployeeIDType> <ID> </ID> </EmployeeIDFilter>		EmployeeIDType and ID occurs for as many Employee ID as to be returned, (see example below)	Possible values for EmployeeIDType are: • Employee_HRISID • LoginID • SSOAuthParam • Email • Guid • LifeSuiteID
<KeyPropertyFilter>		<KeyPropertyFilter> Occurs once per request. Child nodes occur multiple time as applicable to the employee definition.	
<CategoryValue> <CategoryCode/> <CategoryValueCode />			

Node	Required	Description	Notes
<Person> <PersonCode> <EmployeeID> <EmployeeIDType> </EmployeeIDType> <ID> </ID> </PersonCode> </Person>			PersonCode values are defined in Key Properties confiugration. Possible values for EmployeeIDType are: • Employee_HRISID • LoginID • SSOAuthParam • Email • Guid • LifeSuiteID
<Date> <DateCode> </DateCode> <DateRange> <BeginDate/> <EndDate/> </DateRange>			DateCode values are defined in Key Properties confiugration. Date format is YYYY-MM-DD
</KeyPropertyFilter>		close parent node	
<EmployeeStatusFilter> </EmployeeStatusFilter>			Expected Values: • Active • Retired
</FormOwnerFilter>		Close parent node	

Node	Required	Description	Notes
<FormType> </FormType>		FormType can occur many times per call	FormType values match the Form type defined in Onboarding.
<EventFilter>			
<EventID> </EventID> <EventCode> </EventCode>			
<EventStatus> </EventStatus>		When not included eForms from events in all event status will be returned	Expected Values: • In Progress • Complete • Canceled
<CategoryValue> <CategoryCode> </CategoryCode> <CategoryValueCode </CategoryValueCode> </CategoryValue>			
</EventFilter>		close parent node	
</GetCompletedFormsListInput>	✓	close parent node	

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
```

```
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

An XML record describing either the successful task data or a detail of the reason for failure.

The Data record xml is returned as CDATA tag.

If one or more parameters are in error, the ErrorNum returns a value not equal to 1. The XML in the data string describes the errors. The error text in the data string and error codes are described in the "Method Returns" section of each method description.

FormID is in a column called "ObjectId"

Examples

Example 1

The following strXML returns all the forms not yet delivered completed between 11-15-2012 and 12-15-2012:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCompletedFormsListInput>
 <FormFilter>
  <IsFormDelivered>0</IsFormDelivered>
  <CompletionDateRange>
    <BeginDate>2017-11-15</BeginDate>
    <EndDate>2017-12-15</EndDate>
  </CompletionDateRange>
 </FormFilter>
</GetCompletedFormsListInput>
```

Code examples

Sample return data for GetCompletedFormsList

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<EpsStringEx xmlns="http://Eprise">
<ErrorString>No Error</ErrorString>
<ErrorNum>1</ErrorNum>
<Data><![CDATA[<GetCompletedFormsListOutput xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <Forms>
  <Form>
   <ObjectId>12845</ObjectId>
   <FormType>Form_I_9_2009</FormType>
   <Owner>
    <Guid>8e1e0acb-1e46-4d65-9f5e81f71a763727</Guid>
    <HRISID>1000</HRISID>
    <SSOAuthParam />
    <LoginID>Mark.Smith</LoginID>
    <Email>msmith@MySRT.com</Email>
    <LifeSuiteID />
   </Owner>
   <CompletedDate>2012-12-05</CompletedDate>
   <DeliveredDate xsi:nil="true" />
   <Event>
    <EventID>32</EventID>
    <EventCode>Event_7</EventCode>
    <EventName>Onboarding-EVerify</EventName>
    <EventStatus>In Progress</EventStatus>
   </Event>
  </Form>
  <Form>
   <ObjectId>12852</ObjectId>
    <FormType>Form_I_9_2009</FormType>
    <Owner>
    <Guid>c685464d-7055-4c77-9dccbf42361b3aa4</Guid>
    <HRISID>3001</HRISID>
    <SSOAuthParam />
    <LoginID>wes.rogers</LoginID>
    <Email>wrogers@MySRT.com</Email>
    <LifeSuiteID />
   </Owner>
   <CompletedDate>2012-12-06</CompletedDate>
   <DeliveredDate xsi:nil="true" />
   <Event>
    <EventID>35</EventID>
    <EventCode>Event_7</EventCode>
    <EventName>Onboarding-EVerify</EventName>
   </Event>
  </Form>
 </Forms>
</GetCompletedFormsListOutput>
]]>
</Data>
</EpsStringEx>
```

GetCompletedForms

Usage

Retrieves a list of completed form IDs that meet the specified criteria.

Parameters

Parameter	Required	Description	Type
strSecurityToken	✓	Valid Session ID for consuming service	simple string
strStartingDate		Used to specify the start of a date range. The date range queries the completion date of saved forms. A string in yyyy-mm-dd format. Note: The results set will return form IDs greater than the strStartingDate	simple string
strEndingDate		Used to specify the end of a date range. The date range queries the completion date of saved forms. A string in yyyy-mm-dd format. Note: The results set will return form IDs less than the strEndingDate	simple string
strLoginID		Onboarding LogInID of whom the form was completed on behalf of. Specify empty string if you are using the Single Sign On ("SSO") authentication feature. This parameter is mutually exclusive of strFKUserID.	simple string
strFKUserID		If you are using the Single Sign On to identify users, this equates to the <USERID> value in the GetSession request of whom the form was completed. This parameter is mutually exclusive of strLogInID. Specify an empty string if you are specifying a value for strLogInID.	simple string
strFormType		Specify a single Onboarding Form name.	simple string
strDelivered		A method is outlined below to mark a form as "delivered". The flag is be used as a selection criteria. • "" - (empty string) - forms will be returned regardless of their delivered status. • 1 - only forms marked as delivered will be returned • 0 - only forms not marked as delivered will be returned	simple string

Parameter	Required	Description	Type
strEventName		Used to specify the event name associated with the task for the saved forms. "" - (empty string) - forms will be returned regardless of the event. Valid values match the event string as specified on the Event List under Administration -> Manage Events page in the Onboarding user interface.	simple string

Returns

See Returned Array of Forms (see page 135).

To return all completed existing "Federal_W_4" forms:

```
1  Eprise.EpsTableEx t = service.GetCompletedForms(sessionnum,"","","","Federal_W_4","","")
```

GetFormsIDs

Usage

Retrieves a list of (saved) form IDs that meet the specified criteria.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strStartingDate		Used to specify the start of a date range. The date range queries the creation date of saved forms. A string in yyyy-mm-dd format.	simple string
strEndingDate		Used to specify the end of a date range. The date range queries the creation date of saved forms. A string in yyyy-mm-dd format.	simple string
strLogInID		Onboarding LogInID of whom the form was completed on behalf of. Specify empty string if you are using the Single Sign On ("SSO") authentication feature. This parameter is mutually exclusive of strFKUserID.	simple string

Parameter	Required	Description	Type
strFKUserID		If you are using the Single Sign On to identify users, this equates to the <USERID> value in the GetSession request of whom the form was completed on behalf of. This parameter is mutually exclusive of strLogInID. Specify an empty string if you are specifying a value for strLogInID.	simple string
strFormType		Specify a single Onboarding Form name.	simple string
strDelivered		A method is outlined below to mark a form as "delivered". The flag is be used as a selection criteria. • "" - (empty string) - forms will be returned regardless of their delivered status. • 1 - only forms marked as delivered will be returned • 0 - only forms not marked as delivered will be returned	simple string
strVerified		A flag can used as a selection criteria. The Verified flag is set to 1 on a task by completing a task with the "Check if the form is finished upon task completion." option checked. • "" - (empty string) - forms will be returned regardless of their task completion status. • 1 - only forms associated with completed task will be returned • 0 - only forms associated with incomplete tasks will be returned	simple string

Returns

See Returned Array of Forms (see page 135).

Examples

To return all existing forms:

```
1  Eprise.EpsTableEx t = service.GetFormsIDs(sessionnum,"","","","","","","")
```

To return all forms for the user "Jim.Smith":

```
1  Eprise.EpsTableEx t = service.GetFormsIDs(sessionnum,"","","Jim.Smith","","","","")
```

GetCompletedFormsEx

Usage

Retrieves a list of (saved) form IDs that meet the specified filter criteria including event name and category values.

- Use GetFormsIDsEx (see page 133) to retrieve saved Form IDs by creation date
- Use GetCompletedFormsEx to retrieve completed Form IDs by completion date

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strStartingDate		Used to specify the start of a date range. The date range queries the completion date of saved forms. A string in yyyy-mm-dd format.	simple string
strEndingDate		Used to specify the end of a date range. The date range queries the completion date of saved forms. A string in yyyy-mm-dd format.	simple string
strLogInID		Onboarding LogInID or user GUID of whom the form was completed. Specify empty string if you are using the Single Sign On ("SSO") authentication feature. This parameter is mutually exclusive of strFKUserID.	simple string
strFKUserID		If you are using the Single Sign On to identify users, this equates to the <USERID> value in the GetSession request of whom the form was completed. This parameter is mutually exclusive of strLogInID. Specify an empty string if you are specifying a value for strLogInID.	simple string
strFormType		Specify a single Onboarding form type.	simple string
strDelivered		A method is outlined below to mark a form as "delivered". The flag is be used as a selection criteria. • "" - (empty string) - forms will be returned regardless of their delivered status. • 1 - only forms marked as delivered will be returned • 0 - only forms not marked as delivered will be returned	simple string
strEventName		Used to specify the event name associated with the task for the saved forms. "" - (empty string) - forms will be returned regardless of the event. Valid values match the event string as specified on the Event List under Administration -> Manage Events page in the Onboarding user interface.	simple string

Parameter	Required	Description	Type
arrCategoryFilter		Used to specify the benefiter's category values associated with the saved forms. "" - (empty string) - forms will be returned regardless of the associated categories. valid values match the strings as specified on the Manage Hierarchical Category Values under Administration -> Manage Categories page in the Onboarding user interface. See Syntax of the Category Parameter (see page 29).	ArrayOf ArrayOf String

Returns

See Returned Array of Forms (see page 135).

Code examples

To return all existing forms for the category "Old Department" with a value of Marketing

```
String[][] arrCats = new String[1][];

arrCats[0] = new String[]{"Old Department","\\All Departments\\Marketing"};

EpsTableEx tRet = service.GetCompletedFormsIDsEx(sessionnum, "", "", "", "", "", "", "",
 arrCats);
```

To return all the completed forms for employee jane.johnson:

```
EpsTableEx tRet = service.GetCompletedFormsIDsEx(sessionnum, "", "", "jane.johnson", "", "", "",
"", "");
```

GetFormsIDsEx

Usage

Retrieves a list of (saved) form IDs that meet the specified filter criteria including event name and category values.

- Use GetFormsIDsEx to retrieve saved Form IDs by creation date
- Use GetCompletedFormsEx (see page 132) to retrieve completed Form IDs by completion date

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string

Parameter	Required	Description	Type
strStartingDate		Used to specify the start of a date range. The date range queries the creation date of saved forms. A string in yyyy-mm-dd format.	simple string
strEndingDate		Used to specify the end of a date range. The date range queries the creation date of saved forms. A string in yyyy-mm-dd format.	simple string
strLogInID		Onboarding LogInID or user GUID of whom the form was completed. Specify empty string if you are using the Single Sign On ("SSO") authentication feature. This parameter is mutually exclusive of strFKUserID.	simple string
strFKUserID		If you are using the Single Sign On to identify users, this equates to the <USERID> value in the GetSession request of whom the form was completed on behalf of. This parameter is mutually exclusive of strLogInID. Specify an empty string if you are specifying a value for strLogInID.	simple string
strFormType		Specify a single Onboarding form name.	simple string
strDelivered		A method is outlined below to mark a form as "delivered". The flag is be used as a selection criteria. • "" - (empty string) - forms will be returned regardless of their delivered status. • 1 - only forms marked as delivered will be returned • 0 - only forms not marked as delivered will be returned	simple string
strVerified		A flag can used as a selection criteria. The Verified flag is set to 1 on a task by completing a task with the "Check if the form is finished upon task completion." option checked. • "" - (empty string) - forms will be returned regardless of their task completion status. 1 - only forms associated with completed task will be returned • 0 - only forms associated with incomplete tasks will be returned	simple string

Parameter	Required	Description	Type
strEventName		Used to specify the event name associated with the task for the saved forms. "" - (empty string) - forms will be returned regardless of the event. Valid values match the event string as specified on the Event List under Administration>Manage Events page in the Onboarding user interface.	simple string
arrCategoryFilter		Used to specify the benefiter's category values associated with the saved forms. "" - (empty string) - forms will be returned regardless of the associated categories. valid values match the strings as specified on the Manage Hierarchical Category Values under Administration>Manage Categories page in the Onboarding user interface. See Syntax of the Category Parameter (see page 29)	ArrayOf ArrayOf String

Returns

See Returned Array of Forms (see page 135).

Code example

To return all existing forms for the category "Old Department" with a value of Marketing

```
String[][] arrCats = new String[1][];

arrCats[0] = new String[]{"Old Department","\\All Departments\\Marketing"};

EpsTableEx tRet = service.GetFormsIDsEx(sessionnum, "", "", "", "", "", "", "", "",
arrCats);
```

To return all the completed forms for employee jane.johnson:

```
EpsTableEx tRet = service.GetFormsIDsEx(sessionnum, "", "", "jane.johnson", "", "", "", "", "", "")
;
```

Returned Array of Forms

The following methods return the same array of form information:

- GetFormsIDs (see page 130)
- GetFormsIDsEx (see page 133)
- GetCompletedForms (see page 129)
- GetCompletedFormsEx (see page 132)

Returns

A table of type EpsTableEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsTableEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <ArrayOfString>
   <!-- Array of string 'column headers' -->
  </ArrayOfString>
  <ArrayOfString>
   <!-- One array of strings for each result record -->
  </ArrayOfString>
 </Data>
</EpsTableEx >
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

A string array containing columns and their corresponding values. The requested FormID is returned as ObjectId.

Example

```
<EpsTableEx xmlns="http://Eprise">
 <ErrorString />
 <ErrorNum>1</ErrorNum>
<Data>
 <ArrayOfString>
  <string>ObjectId</string>
  <string>Form_Type</string>
  <string>Owner</string>
  <string>CompletedDate</string>
  <string>Delivered</string>
  <string>Name</string>
  <string>EventID</string>
 </ArrayOfString>
 <ArrayOfString>
  <string>10544</string>
  <string>Form_I_9_2009</string>
```

```
   <string>eace7ee3-d566-432f-b32fe0fa48ce1df3</string>
   <string>2011-06-21</string>
   <string />
   <string>Onboarding-Short</string>
   <string>85</string>
  </ArrayOfString>
 </Data>
 </EpsTableEx>
```

Return Details:

Return Value	Description
ObjectId	The ID of the Form (strFormID) that can be used in GetFormXML (see page 137), GetFormPDF (see page 139), MarkFormDelivered (see page 145)and DeleteForm (see page 148).
Form_Type	Indicates which form template was used for the form.
Owner	The Guid of the event benefiter of the event the form was completed in.
CompletedDate	The date (YYYY-MM-DD) the form was completed in the system.
Delivered	The date (YYYY-MM-DD) the form was marked delivered with the MarkFormDelivered (see page 145) method.
Name	The display name of the form.
EventID	The User Event ID that the form was completed in, this can be used to link forms owned by the same user to a single event.

GetFormXML

Usage

Retrieves an existing XML form based on the specified FormID.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string

Parameter	Required	Description	Type
strFormID	☐	A single formID . FormID is returned by GetFormsIDs, GetFormsIDsEX or GetCompletedFormsIDEx method as <ObjectID>	simple string

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

A string containing the XML of the requested FormID

Code examples

To return XML of the GetFormIDs method and write the output a file:

```
Eprise.EpsStringEx ret = service.GetFormXML(sessionnum,t.Data[i][nObjectid]); if
 (ret.ErrorNum != 1)
{
  Console.Out.WriteLine("Error while trying to retrieve xml for " + t.Data\[i\]\
[nObjectid\] + " Error:" + t.ErrorString);
  return;
}

```

```
String strXML = ret.Data; String strFileName = String.Format("c:\\downloadedforms\
\Form{0}.xml",t.Data[i][nObjectid]);
FileStream f = File.OpenWrite(strFileName);
Byte [] arrBytes = ue.GetBytes(strXML);
f.Write(arrBytes,0,arrBytes.Length);
f.Close();
```

To return XML of the FormID "9678":

```
Eprise.EpsStringEx ret = service.GetFormXML(sessionnum,"9678");
```

Example output of the Data string ("ret.Data") for a form named TEST_W4 with five fields named SSN, City, State, Zip, and DT_NOW:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<response>
 <forms>
  <form ObjectId="8264" Form_Name="TEST_W4">
   <SSN>123-456-789</SSN>
   <City>Clinton</City>
   <State>MA</State>
   <Zip>01510</Zip>
   <DT_NOW>2006-08-02</DT_NOW>
  </form>
 </forms>
</response>
```

GetFormPDF

Retrieve populated PDF files

Onboarding custom eforms can be designed to merge the form data to an existing PDF file format. All eForm Builder eForms have a corresponding PDF file.

Usage

Used to request form data be returned as a PDF file. The form data is merged with a (pre-configured) PDF file based on the FormID.

The user obtaining the session ID must be able to access the form and task (the form is associated with). The programmatic user requires View All Tasks and Manage Forms permission.

Input

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string

Parameter	Required	Description	Type
strFormID	☐	A single formID	simple string

Returns

A binary string of type EpsBinaryEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsBinaryEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[Binary String]]>
 </Data>
</EpsBinaryEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Binary data (PDF binary)

ⓘ Note: The configuration of forms to map data to a designated PDF file is a service provided by SilkRoad Support.

Code example

Retrieve the corresponding PDF based on the given FormID. Write the PDF to a "downloadedforms" directory:

```
Eprise.EpsBinaryEx ret = service.GetFormPDF(sessionnum,t.Data[i][nObjectid]); if
 (ret.ErrorNum != 1)
{
 Console.Out.WriteLine("Error while trying to retrieve pdf for " + t.Data[i][nObjectid]
+ " Error:" + t.ErrorString);
 return;
}

Byte[] arrBytes = ret.Data; String strFileName =
 String.Format("c:\\downloadedforms\\Form{0}.pdf",t.Data[i][nObjectid]);

```

```
10
11  FileStream f = File.OpenWrite(strFileName);
12  f.Write(arrBytes,0,arrBytes.Length);
13  f.Close();
```

GetFormPDFEx

Usage

Retrieves an archival compliant PDF. The supported archival PDF formats are:

- PDF/A-1 Basic and Accessible
- PDF/A-2 Basic, Accessible and Unicode
- PDF/A-3 Basic, Accessible and Unicode

The PDF Association has an overview of the PDF/A standard at https://www.pdfa.org/resource/pdfa-in-a-nutshell-2-0/.

Input

Parameter	Required	Description	Type
strSecurityToken	✓	Valid Session ID for consuming service	simple string
strFormID	✓	A single formID	simple string
strPDFStandard	✓	The PDF/A standard that the eForm PDF is to be converted to. Valid Values: (case sensitive) • Level1A for PDF/A-1 Accessible • Level1B for PDF/A-1 Basic • Level2A for PDF/A-2 Accessible • Level2B for PDF/A-2 Basic • Level2U for PDF/A-2 Unicode • Level3A for PDF/A-3 Accessible • Level3B for PDF/A-3 Basic • Level3U for PDF/A-3 Unicode	simple string

Returns

A binary string of type EpsBinaryEx.

An XML document defined as:

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <EpsBinaryEx xmlns="http://Eprise">
3   <ErrorString>No Error</ErrorString>
4   <ErrorNum>1</ErrorNum>
5   <Data>
```

```
    <![CDATA[Binary String]]>
  </Data>
</EpsBinaryEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Binary data (PDF binary)

GetFormPDFEx Error messages

<Type>	<Message>	<ErrorCode>	Description
Conversion Failure	Conversion of the PDF for {0} to {1} failed		• 0 = eForm PDF Name • 1 = Conversion standard attempted `Conversion of the PDF for Federal_W_4_Version_18.pdf to Level1B failed`

GetFormI9Package

Usage

Retrieves a zip file including Form I-9 related forms and documents:

- The PDF of each Form I-9 stored on the employee profile.
- The Audit Trail for each Form I-9 included.
- Any Form I-9 uploaded documents.
- E-Verify case details for each E-Verify case associated with the employee.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

Example Structure of GetUserProfileEx2 strXML

```
<?xml version="1.0" encoding="utf-8"?>
<GetFormI9PackageInput>
  <EmployeeIDFilter>
    <EmployeeIDType>LoginID</EmployeeIDType>
    <ID>Tammy.Jones</ID>
  </EmployeeIDFilter>
</GetFormI9PackageInput>
```

Nodes

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	☐	XML Document header	
<GetFormI9PackageInput>	☐	Parent node	
<EmployeeIDFilter>	☐	Parent node	
<EmployeeIDType></EmployeeIDType>	☐	ID type	Acceptable values are: • Employee_HRISID • LoginID • SSOAuthParam • Email • Guid • LifeSuiteID See "Employee ID Types" (see page 106) for descriptions. example:<EmployeeIDType>LoginID</EmployeeIDType>

Node	Required	Description	Notes
<ID></ID>	☐	ID value	Specific to unique identifier for each user. This value is coupled to the Employee ID Type specified. Example: <ID>Tammy.Jones</ID>
</EmployeeIDFilter>	☐	close node	
</GetFormI9PackageInput>	☐	close parent node	

Returns

A binary string of type EpsBinaryEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsBinaryEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[Binary String]]>
 </Data>
</EpsBinaryEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Binary Data (Zip file of the Employee's I-9 Information.)

GetFormI9Package Error messages

<Type>	<Message>	<ErrorCode>	Description
UserError	The LoginId ({0}) used to identify a user is used by more than one user	31059	
UserError	No user could be found for the id: {0}	31060	
	There are no I9 Documents to download for this user.		If there are not documents to return the method does not return any data.

MarkFormDelivered

Usage

A delivery status flag can be set based on the processing specifications of your eForms.

Used to indicate the delivery status of a form based on the FormID.

Note: After the form delivery status is set to completed, the task associated with the form can not be reopened.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strFormID	☐	A single formID	simple string
strDelivered	☐	Use: 0 - To set a form delivery status to undelivered. After a form is completed the delivered state is set to 0. 1 - To set a form to delivery status to delivered. By default, delivered is also locked. The associated task can not be reopened and no form updates are allowed and the eForm cannot be deleted (see DeleteForm (see page 148)) 2 - To set a form to delivery status to delivered and unlocked. Unlocked forms allow the task owner to reopen the associated task and update the form. To delete eForms after they are delivered they must be delivered unlocked. (see DeleteForm (see page 148))	simple string

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Empty string

Examples

```
Eprise.EpsStringEx ret = service.MarkFormDelivered(sessionnum,t.Data[i][nObjectid],"1");

if (ret.ErrorNum != 1)

{
 Console.Out.WriteLine("Error while trying to mark form " + t.Data[i][nObjectid]+ " Error:" +
t.ErrorString);
 return;
}
```

Implantation considerations

After processing a form, if you choose to set the delivery status to "1", the task with the associated eForm can not be reopened. This functionality ensures the data is not updated after an e-form is delivered to the external application. If the task is attempted to be reopened, the following message is displayed.

After processing a form, if you choose to set the delivery status to "2", the task with the associated eForm can be reopened and the form can be updated. If the system accepting the forms data processes duplicates, this is the recommended technique. For example Jane Doe completes a form. Your integration logic retrieves the completed form and updates your external system. If Jane later updates the form, the delivered state is set back to 0 (zero). if you want the updated information to be updated to your external system, use this setting. This functionality ensures form data can be updated after it is delivered.

Based on the requirements of the dependent application for eForm data changing after it has been delivered, you can implement functionality to (re)set the strDelivered flag status to incomplete for *specific* situations.

It is not valid to change the strDelivered flag status from "1" (delivered / locked) to "2" (delivered / unlocked).

- A valid state transition for the strDelivered flag is "1" (delivered / locked) to "0" (undelivered).
- A valid state transition for the strDelivered flag is "0" (undelivered) to "2" (delivered / unlocked).

This is important if you have forms in a locked status that you want to unlock. You must reset the status to "0" before setting it to "2".

Special processing for unlocked I-9 Form data

E-Verify is very specific with its rules regarding one active E-Verify case per employee. Onboarding has multiple checks to prevent an employer from opening multiple E-Verify cases at the same time. In support of this, updating I-9 Form data that is associated with an E-Verify case, by default is not permitted. The default behavior is to disallow re-opening a closed E-Verify submission task.
Procedure for updating I-9 forms after they have been marked as delivered to an external system.

In 2.7.x the process flow (for an I-9 form) is:

1. Form is created via step 1 task. It is editable only by the benefiter of the event.
2. Form is updated via a step 2 task. It is editable by the task owner or I-9 admin. Form is completed and step 2 task is completed.
3. Form data is fed to E-verify submission screen. Step 3 task is completed when E-Verify case is closed.
4. Closed step 3 tasks (step 2 and step 1) tasks cannot be reopened.
5. Form is selected, form data is retrieved and passed to external system
6. Form is marked as delivered by calling MarkFormAsDelivered (FormID, 1). 1 notes delivered and locked.
7. No I-9 task can be reopened for a locked form. No form updates are allowed.

In 2.8x, and later, the process flow (for an editable closed I-9 form) is:

1. Request your configuration be upgraded to allow for a step 1 or step 2 task to be opened without affecting the other tasks in the workflow.
    - Set "Enable Always Reopen Dependent Tasks checkbox for I-9 Step 1 and 2 task definitions" to On.
    - Recycle
    - Edit the step 1 and 2 task definitions to uncheck the "Always Reopen Dependent Tasks" checkbox.
    - Confirm you have selected "Restrict re-opening of task to Event Coordinator:"
2. For new employees launched:

- Form is created via step 1 task. It is editable only by the benefiter of the event.
- Form is updated via a step 2 task. It is editable by the task owner or I-9 admin. Form is completed and step 2 task is completed.
- Form data is fed to E-verify submission screen. Step 3 task is completed when E-Verify case is closed.
- Closed step 3 tasks (step 2 and step 1) tasks are now allowed to be reopened (independently of each other).

3. I-9 Forms are selected, form data is retrieved (and used to update your external system)
4. Form is marked as delivered by calling MarkFormAsDelivered (FormID, "2"). 2 notes delivered and unlocked.
5. I-9 tasks can be reopened for a unlocked form.
    - Section 1 can be updated via step 1 task. It is editable only by an I-9 Administrator or the benefiter once it is reopened.
    - Section 2 can be updated via step 2 task. It is editable by the task owner or I-9 admin. Form is completed and step 2 task is completed.
    - Step 2 task continues to be completed via "Complete & E-Verify" button. **The employee will not be submitted to E-Verify again. It simply re-displays the E-Verify status**.

In 2.8x the process flow (for an I-9 form completed in RC 2.7.x) is:

1. Request to configure the instance of Onboarding to allow for a step 1 or step 2 task to be opened without affecting the other tasks in the workflow.
2. Set "Enable Always Reopen Dependent Tasks checkbox for I-9 Step 1 and 2 task definitions" to On.
3. Edit the step 1 and 2 task definitions to uncheck the "Always Reopen Dependent Tasks" checkbox
4. Form was marked as delivered by calling MarkFormAsDelivered (FormID, "1") during the 2.7.x cycle. It is now delivered / locked.
5. Reset the form to unlocked using:
    - **NOTE**: If a form is currently marked as delivered, it must be reset to undelivered before it can be unlocked.
    - Form is reset by marking the form as undelivered by calling MarkFormAsDelivered (FormID, "0"). "0" notes undelivered.
    - Form is marked as delivered but unlocked by calling MarkFormAsDelivered (FormID, "2"). "2" set the form to delivered and unlocked.
6. I-9 tasks can be reopened for a unlocked form.
    - Section 1 can be updated via step 1 task. It is editable only by an I-9 Administrator or benefiter once it is reopened.
    - Section 2 can be updated via step 2 task. It is editable by the task owner or I-9 admin. Form is completed and step 2 task is completed.
    - Step 2 task continues to be completed via "Complete & E-Verify" button. **The form will not be submitted to E-Verify again. It simply re- displays the e-verify status**.
    - On completing the form the delivery status is set to "0" (undelivered).

NOTE: Task notes are updated to state the task was re-opened and closed. We recommend you update the "margin notes" text box on the I-9 form as well as add appropriate task notes. All versions of the I-9 form are available via the auditor's portal.

DeleteForm

Usage

Used to delete a form from Onboarding based on a FormID.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strFormID	☐	A single formID	simple string

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Empty string

Examples

To delete a specified RedCarpet form:

```
Eprise.EpsStringEx ret = service.DeleteForm(sessionnum,t.Data[i][nObjectid]); if
 (ret.ErrorNum != 1)
{
 Console.Out.WriteLine("Error while trying to delete " + t.Data[i][nObjectid] + "
Error:" + t.ErrorString);
```

```
4
5   return;
6  }
```

Considerations

To clear data from Onboarding, the DeleteForm API can be used; however, when an eForm is delivered (locked / strDelivered ="1") the eForm is read-only and cannot be deleted. To delete eForms one of two workflows must be used:

Option 1:

1. Use MarkFormDelivered (see page 145)with strDelivered = "2"
2. Use DeleteForm the form when needed.

Option 2:

1. Use MarkFormDelivered (see page 145)with strDelivered = "1" when collecting the eForm data to ensure the eForm workflwo cannot be re-opened.
2. When the eForm needs to be deleted, use MarkFormDelivered with strDelivered ="0" to undeliver the eForm
3. Use DeleteForm to delete the form that is now undelivered.

Documents API

Onboarding web methods provide an option to retrieve uploaded employee documents:

- Retrieve uploaded documents accessible from the Edit Employee tab pages
- Delete existing documents

GetUploadedDocumentList

Usage

GetUploadedDocumentList returns a list of document IDs. Documents returned include supporting documents uploaded for I-9s, employee documents (uploaded by the employee through the portal) and documents uploaded by the employer through the Documents tab. The document ID list is created based on specified search criteria.

If search criteria is specified that does not match any data, no data is returned.

The document ID list can be used as input to the GetUploadedDocument method with returns the binary data of the document.

Input

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

Nodes in GetUploadedDocumentList strXML

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	✓	XML Document header	
<GetUploadedDocumentListInput>	✓	Parent node occurs once	
<UploadDateRange> <BeginDate></BeginDate> <EndDate></EndDate> </UploadDateRange>		Optional - date range of upload	Begin Date and / or End Date can be specified. Format is YYYY-MM-DD

Node	Required	Description	Notes
<DocumentOwnerFilter> <EmployeeIDFilter> <EmployeeIDType> </EmployeeIDType> <ID> </ID> </EmployeeIDFilter> </DocumentOwnerFilter>		EmployeeIDType and ID occurs for as many Employee ID as to be returned, (see example below)	Possible values for EmployeeIDType are: • Employee_HRISID • LoginID • SSOAuthParam • Email • Guid • LifeSuiteID
<KeyPropertyFilter>		<KeyPropertyFilter> Occurs once per request. Child nodes occur multiple time as applicable to the employee definition.	
<CategoryValue> <CategoryCode/> <CategoryValueCode />			

Node	Required	Description	Notes
<Person> <PersonCode> <EmployeeID> <EmployeeIDType> </EmployeeIDType> <ID> </ID> </PersonCode> </Person>			PersonCode values are defined in Key Properties configuration. Possible values for EmployeeIDType are: • Employee_HRISID • LoginID • SSOAuthParam • Email • Guid • LifeSuiteID
<Date> <DateCode> <DateRange> <BeginDate/> <EndDate/> </DateRange> </DateCode>			DateCode values are defined in Key Properties configuration. Date format is YYYY-MM-DD
</KeyPropertyFilter>		close parent node	

Node	Required	Description	Notes
<DocumentTypeFilter> <DocumentType> </DocumentType> </DocumentTypeFilter/>		Optional. Document TypeFilter occurs once per call	DocumentType values are: • DocumentUpload • EmployeeDocumentUpload • I9Upload
</GetUploadedDocumentListInput>		close parent node	

Sample strXML

To get all uploaded documents:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetUploadedDocumentListInput>
  <DocumentTypeFilter>
    <DocumentType>DocumentUpload</DocumentType>
  </DocumentTypeFilter>
</GetUploadedDocumentListInput>
```

To get all documents uploaded in support of I-9s by begin and end date:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetUploadedDocumentListInput>
  <UploadDateRange>
    <BeginDate>2010-11-04</BeginDate>
    <EndDate>2012-11-04</EndDate>
  </UploadDateRange>
  <DocumentTypeFilter>
    <DocumentType>I9Upload</DocumentType>
  </DocumentTypeFilter>
</GetUploadedDocumentListInput>
```

To get all documents uploaded in support of I-9s by end date:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetUploadedDocumentListInput>
  <UploadDateRange>
    <EndDate>2012-12-01</EndDate>
  </UploadDateRange>
  <DocumentOwnerFilter>
  </DocumentOwnerFilter>
  <DocumentTypeFilter>
    <DocumentType>I9Upload</DocumentType>
  </DocumentTypeFilter>
</GetUploadedDocumentListInput>
```

To get all documents for multiple users by user GUID:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetUploadedDocumentListInput>
  <DocumentOwnerFilter>
    <EmployeeIDFilter>
      <EmployeeIDType>Guid</EmployeeIDType>
      <ID>c3c012fb-b2db-4f13-aafd2ac3d08369ae</ID>
      <ID>e82bb984-79c5-4de3-adbab5bf4c53faf7</ID>
    </EmployeeIDFilter>
  </DocumentOwnerFilter>
</GetUploadedDocumentListInput>
```

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

String containing CDATA with PageID as described below

NOTE: the output of this method has been modified to include document CreateDate in release 2016.1. If you would like to have CreateDate included in
the output, you must remove the Transformation added on upgrade. See Chapter 7 for information on Managing Transformations.

Sample Output

The following sample output returns three uploaded documents:

```
<EpsStringEx xmlns=http://Eprise/]
<ErrorString>No Error</ErrorString>
<ErrorNum>1</ErrorNum>
<Data><![CDATA[[
  <Documents xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
    <Document>
      <PageId>75d515d0-281f-4cc1-a8c5df12a2d72648</PageId>
      <Type>pdf</Type>
      <DocumentType>DocumentUpload</DocumentType>
      <DocumentTitle>FormAddDoc_Drug</DocumentTitle>
      <CreateDate>2010-09-02</CreateDate>
      <Owner>
        <Guid>6e73c680-5603-4914-82e5be4a5057a7a5</Guid>
        <Employee_HRISID />
        <SSOAuthParam />
        <LoginID>I9_EV_EC5</LoginID>
        <Email>I9_EV_EC5@e.com</Email>
        <LifeSuiteID />
      </Owner>
    </Document>
    <Document>
      <PageId>605d9257-cdbd-4c1f-9c9dab344c02ae28</PageId>
      <Type>pdf</Type>
      <DocumentType>DocumentUpload</DocumentType>
      <DocumentTitle>FormAddDoc_Conf</DocumentTitle>
      <CreateDate>2010-09-02</CreateDate>
      <Owner>
```

```
        <Guid>6e73c680-5603-4914-82e5be4a5057a7a5</Guid>
        <Employee_HRISID />
        <SSOAuthParam />
        <LoginID>I9_EV_EC5</LoginID>
        <Email>I9_EV_EC5@e.com</Email>
        <LifeSuiteID />
      </Owner>
    </Document>
    <Document>
      <PageId>ca1bcab3-20c3-4874-8e2aa2c23445e499</PageId>
      <Type>pdf</Type>
      <DocumentType>DocumentUpload</DocumentType>
      <DocumentTitle>Form Additional Doc Conf</DocumentTitle>
      <CreateDate>2010-01-24</CreateDate>
      <Owner>
        <Guid>fe61a69b-bec8-4a44-9688d361486e45a8</Guid>
        <Employee_HRISID />
        <SSOAuthParam />
        <LoginID>TwoEVInPro</LoginID>
        <Email>TwoEVInPro@e.com</Email>
        <LifeSuiteID />
      </Owner>
    </Document>
  </Documents>]]>
</Data>
</EpsStringEx>
```

GetUploadedDocument

Usage

Similar to the GetFormPDF method, this method returns binary data for the specified uploaded document. It is used with GetUploadedDocumentList.

Input

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

Returns

A binary string of type EpsBinaryEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsBinaryEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[Binary String]]>
 </Data>
</EpsBinaryEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Binary content for the specified uploaded document.

DeleteUploadedDocument

Delete an uploaded document

A delete method for your uploaded documents based on your specifications.

DeleteUploadedDocument

Usage

Used to delete a document from Onboarding based on a document ID. A company may choose to delete The document ID is referred to as "PageID" and is returned by GetUploadedDocumentList method.
The API user used to establish the session must have the following privileges assigned to their team (to insure all applicable documents can be deleted)

- I-9 Processor
- View Employee Documents
- View Forms
- Edit All Users

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strPageID	☐	page Id (GUID) returned by GetUploadedDocumentList	simple string

Return

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

String with document name, who deleted the file, and when.

```
<EpsStringEx>
<ErrorString>No Error</ErrorString>
<ErrorNum>1</ErrorNum>
<Data>Script Error.JPG was deleted by Admin on 4/7/2015 10:33:52 PM</Data>
</EpsStringEx>
```

Example

Code example

To delete a specified RedCarpet document:

```
Eprise.EpsStringEx ret = service.DeleteUploadedDocument(sessionnum,t.Data\[i\]\
[nObjectid\]);

if (ret.ErrorNum != 1)
{
 Console.Out.WriteLine("Error while trying to delete " + t.Data\[i\]\[nObjectid\] + "
Error:" + t.ErrorString);
 return;
}
```

UploadEmployeeDocument

Usage

Upload a document to an Employee Profile. This can be used to upload either an employer-uploaded document or an I-9 uploaded document.

Prerequisite(s): You have to edit the ceForm element related to the style that you'll be using for the upload. The element will require additional fields for the attributes to work properly.

In the examples below, you would have to add the text field of 'DocumentType' to the element. Path to the element in Eprise: /main/RedCarpet/FormTemplates/Document_Upload/ceForm

Warning(s): Do not use comments within your code for this method.

Parameters

Parameter	Required	Description	Type
strSecurityToken	✓	Valid Session ID for consuming service	simple string
strUserId	✓	Used to identify the user to attach the document to. Valid values are: • LoginID • AuthParam • Email Address • UserGUID	simple string
strDocumentStyleName	✓	Used to identify the type of document that will be uploaded. Supported options: • Document_Upload • I_9_Upload	simple string

Parameter	Required	Description	Type
strFileName	✓	The name of the file that is being uploaded	simple string
strBase64Data	✓	Base 64 data of the file being uploaded	Base 64 encoded
arrFormFields		And array of the document attributes. <epr:arrFormFields> <epr:ArrayOfString> <epr:string>DocumentType</epr:string> <epr:string>Resume</epr:string> <epr:ArrayOfString> </epr:arrFormFields>	Array of Strings

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Error messages

<Type>	<Message>	<ErrorCode>	Description
Error	No user could be found for the id: {0}	31060	
Error	Missing required information: (strBase64Data or strUserID missing.)	34000	
Error	Insufficient privileges	34001	
Error	Invalid input: strDocumentStyleName must be 'Document_Upload' or 'I_9_Upload'	42000	

Examples

The below example includes just a document that will be uploaded.

Note: The base 64 data is truncated for this example.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:epr="http://
Eprise">
   <soapenv:Header/>
   <soapenv:Body>
      <epr:UploadEmployeeDocument>
         <epr:strSecurityToken>18286683161789914795114703238921266702</epr:strSecurityToken>
         <epr:strUserId>Jane.Smith</epr:strUserId>
         <epr:strDocumentStyleName>Document_Upload</epr:strDocumentStyleName>
         <epr:strFileName>MakeShift_Fields5.xlsx</epr:strFileName>
         <epr:strBase64Data>UEsDBBQACAgIAIlMglAAAAAAAAAAAAAAAARAAAAZG9jUHJvcHMvY29yZS54bWytkc...<
/epr:strBase64Data>
      </epr:UploadEmployeeDocument>
   </soapenv:Body>
</soapenv:Envelope>
```

This example is an uploaded document that includes an attribute named "DocumentType".

Note: The base 64 data is truncated for this example.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:epr="http://
Eprise">
   <soapenv:Header/>
   <soapenv:Body>
      <epr:UploadEmployeeDocument>
         <epr:strSecurityToken>18286683161789914795114703238921266702</epr:strSecurityToken>
         <epr:strUserId>Jane.Smith</epr:strUserId>
```

```
         <epr:strDocumentStyleName>Document_Upload</epr:strDocumentStyleName>
         <epr:strFileName>JaneSmithResume.docx</epr:strFileName>
         <epr:strBase64Data>UEsDBBQACAgIAIlMglAAAAAAAAAAAAAAAAARAAAAZG9jUHJvcHMvY29yZS54bWytkc...<
/epr:strBase64Data>
         <epr:arrFormFields>
                <epr:ArrayOfString>
                <epr:string>DocumentType</epr:string>
                <epr:string>Resume</epr:string>
            <epr:ArrayOfString>
         </epr:arrFormFields>
      </epr:UploadEmployeeDocument>
   </soapenv:Body>
</soapenv:Envelope>
```

Event APIs

You can use event and task APIs to create and edit employee events. The task APIs allow tasks to be added and deleted from existing events. In all cases of executing the event and task methods, the event and task definition must be predefined (through the administrative pages of the Onboarding interface).

The available methods to manage employee events are:

- Launch a new event for an existing employee
- Edit an existing LifeCycle event

The event definition must be defined through the Onboarding user interface. Fields defined in the event definition determine the attributes of the event record passed to the Events methods.

Input specifications

Both of the event methods use two arguments.

1. A Session ID as described in "Logging in to Consume the Service".
    - The Onboarding user executing the Events methods must be a member of the Event Coordinator team for each event included in the Events record.
2. An XML document containing one or more events. Each Event in the XML identifies the employee benefiting from the event, and the field values of the event.
    - The event record, called "xmlEventData", uses the same syntax for each method. The syntax is described below.
    - The requirements of the LaunchEvent method differ from the UpdateEvent method. The rules are outlined with each event.

Note: SessionID for event methods must include event coordinator membership.

Field descriptions of the xmlEventData

The second parameter of each Event method is an XML document passed as a simple string. The string contains an events record with the following general format:

- The root element is <Events>.
- The parent element, <Event> occurs once for each event to be launched. The <Event> record is composed of two parts:
    a. The child element must contain one of the unique user identifiers described below.
    b. Child and subchild elements to populate or update the event fields as defined on the event definitions. The syntax to match the event definition is described below.

Details of the <Event> Input record

Two parts of the <LaunchedEvent> input record are:

Part One: Element to uniquely identify an employee to benefit from the event or the event update

You can choose one from three techniques to uniquely identify an employee for LaunchEvent and UpdateEvent. Three techniques are available for backward compatibility.

1. Four elements are provided to choose from. The naming convention of elements used for unique identifiers are prefixed with "ForWhom_". The record should contain *only one* of these options. If you are using an external application to authenticate your users, SilkRoad recommends you use ForWhom_AuthParam to identify each user.

"ForWhom" Identifier	Description
ForWhom_LoginId	Onboarding LoginId for employees using Standard Authentication
ForWhom_AuthParam	LoginID for employees using External Authentication
ForWhom_Email	employee Email address if the employee email address is
ForWhom_Guid	Onboarding creates and stores a GUID for each employee. This value is returned from GetUserProfileEx method.unique.

2. You can use this flexible technique allowing your to set the identifier type by specifying it as a value, where EmployeeIDType is on of the values described in Employee ID Types (see page 20).

```
<ForWhom_EmployeeID>
 <EmployeeIDType></EmployeeIDType>
 <ID></ID>
</ForWhom_EmployeeID>
```

3. LaunchEvent and UpdateEvent also accept this syntax, where EmployeeIDType is on of the values described in Employee ID Types (see page 20).

```
<EmployeeID>
 <EmployeeIDType></EmployeeIDType>
 <ID></ID>
<EmployeeID>
```

Note: f you want to identify the user by the LifeSuiteID or the EmployeeHRISID your must use either the second or thrid technique.

Part Two: Named event to be launched or updated its corresponding event field(s)

The naming convention of the sub-child element tags is determined by the fields defined in the event definition (in the Onboarding user interface).

- Each defined category field uses a Name / Value / Type syntax.
- Each defined non-category field uses a Name / Value pair syntax

When generating your event syntax note that Event categories are populated with a **single leaf node value** for each category. The event categories are child nodes of the <Event> element. Other than the parent element, the same syntax is used for the Event as used in the user import syntax.

Category Syntax:

The syntax rules for Event Categories are:

1. Each Category is coupled with three child elements:
   a. <Name>
   b. <Value>
   c. <Type>

2. Category Values, each value is listed for the corresponding "Value" node in the XML syntax.

XML Input Nodes

Node	Required	Description
<Events>	☐	Parent Node
<Event>	☐	Parent Node

<Category>
<Name>
<Value>
<Type>
</Category>
	☐	Occurs for each category defined on the event
		See: Category Syntax (see page 29).
		Example:
		`<Name>Location</Name>`
		`<Value>MA_BEDFORD</Value>`
		`<Type>Code</Type>`
		`<Name>Location</Name>`
		`<Value>BEDFORD</Value>`
		`<Type>Name</Type>`

<Person>
<Name>
<Value>
</Person>
	☐	Occurs for each person defined on the event
		See: Person Syntax.
		`<Name>HR Coordinator</Name>`
		`<Value>Amy.HRCoord</Value>`

<Date>
<Name>
<Value>
</Date>
	☐	Occurs for each date defined on the event
		See: Date Syntax.
		`<Name>Start</Name>`
		`<Value>2014-10-31</Value>`
		Date Format: YYYY-MM-DD

Output specifications

Similar to the BulkImport and XMLEditUser methods, each method returns an XML document defined as:

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Node returned with descriptive XML containing. If the ErrorNum is zero, the Date node will include an ErrorCode.

```
<?xml version="1.0" encoding="UTF-8"?>
<EpsStringEx>
 <ErrorString>Error Launching Events.</ErrorString>
 <ErrorNum>0</ErrorNum>
 <Data>
  <LaunchEventResults>
   <Note>0 events launched</Note>
   <Note>1 events not launched</Note>
   <Note>Parsing started at Wednesday, March 04, 2014 1:24:31 PM and ended at Wednesday, March 04,
2014 1:24:32 PM</Note>
   <EventError>
    <LoginID>Terri.Newa</LoginID>
    <Name>I-9 and E-Verify</Name>
    <Error>
     <Type>Event Error</Type>
     <Message>Unknown event name</Message>
     <ErrorCode>30003</ErrorCode>
    </Error>
   </EventError>
   <BadEvents>
    <Event>
```

```
      <ForWhom_EmployeeID>
       <EmployeeIDType>LoginID</EmployeeIDType>
       <ID>Terri.Newa</ID>
      </ForWhom_EmployeeID>
      <Name>I-9 and E-Verify</Name>
      <Person>
       <Name>I-9 Coordinator</Name>
       <Value>Amy.HRCoord</Value>
      </Person>
      <Date>
       <Name>Start</Name>
       <Value>2014-10-31</Value>
      </Date>
      <Category>
       <Name>Work Location</Name>
       <Value>MA</Value>
       <Type>Code</Type>
      </Category>
     </Event>
    </BadEvents>
   </LaunchEventResults>
  </Data>
 </EpsStringEx>
```

Finding the ErrorCode

The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.

If the ErrorNum returns a 1 the ErrorString is equal to "No Errors".

If the ErrorNum returns a value not equal to 1 the ErrorString will return "There were errors" for older methods and a descriptive string for newer methods. For information on the Error, the Data node consistently includes an Error node with the ErrorCode. When the ErrorNum is not equal one, the error(s) are described in the Data string.

Data

An XML record describing either the successful event modifications or a detail of the reason for failure.

If one <Event> record or more is in error, the ErrorNum returns a value not equal to 1. The XML in the Data string will describe the errors. The errors are outlined with each event.

LaunchEvent

Usage

Used to launch an event for an existing employee.

Parameters

Parameter	Required	Description	Type
strSecurityToken	✓	Valid Session ID for consuming service	simple string
xmlEventData	✓	XML record formatted following the rules outlined in "Field Descriptions of the xmlEventData (see page 164)".	simple string

The LaunchEvent method has the same requirements as launching an event through the Onboarding interface. The requirements are:

1. All fields defined in the event definition are required in the <Event> record.
2. An employee is limited to a single active event of an event type.
3. An employee must be active to have an event launched on their behalf.
4. Tasks and notifications are generated for the employee at the time the event is successfully created.
5. The successful LaunchEvent includes an EventID in the return data. The EventID is a unique identifier for this instance of the event. The EventID is unique to this employee for this event.

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Data Node returns a node called <LaunchEventResults>

Note: If you started using SilkRoad Onboarding before version 2015.1, on upgrade an XSLT will be applied to your site. To obtain the most recent output, go to the Manage Transformation page and delete the LaunchEvent (and/or) UpdateEvent transformation. See Using Transformations (see page 264) for more information on the transformations.

Examples

Example "xmlEventData" input:

```
<?xml version="1.0" encoding="UTF-8"?>
<Events>
 <Event>
  <ForWhom_EmployeeID>
   <EmployeeIDType>LoginID</EmployeeIDType>
   <ID>Terri.Newa</ID>
  </ForWhom_EmployeeID>
  <Name>I-9 and E-Verify</Name>
  <Person>
   <Name>I-9 Coordinator</Name>
   <Value>Amy.HRCoord</Value>
  </Person>
    <Person>
        <Name>Manager</Name>
        <EmployeeID>
            <ID>888</ID>
            <EmployeeIDType>Employee_HRISID</EmployeeIDType>
        </EmployeeID>
    </Person>
  <Date>
   <Name>Start</Name>
   <Value>2014-10-31</Value>
  </Date>
  <Category>
   <Name>Work Location</Name>
   <Value>MA</Value>
   <Type>Code</Type>
  </Category>
 </Event>
</Events>
```

Sample Output

```
<EpsStringEx>
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <LaunchEventResults>
   <Note>1 events launched</Note>
   <LaunchedEvents>
```

```
    <LaunchedEvent>
     <ForWhomEmployee>
      <Guid>37c61606-af88-4f6e-a57e9c968a33f1d6</Guid>
      <Employee_HRISID>TNewa</Employee_HRISID>
      <SSOAuthParam>TNewa</SSOAuthParam>
      <LoginID>Terri.Newa</LoginID>
      <Email>tnewa@some.com</Email>
      <LifeSuiteID>50f64ae6-1563-4c22-9995-330cbba21641</LifeSuiteID>
      <FirstName>Terri</FirstName>
      <LastName>Newa</LastName>
      <MiddleName></MiddleName>
      <MiddleInitial></MiddleInitial>
     </ForWhomEmployee>
     <Name>I-9 and E-Verify</Name>
     <EventID>4</EventID>
     <EventStatus>Active</EventStatus>
     <CreationDate>2014-11-11T21:19:55</CreationDate>
     <EffectiveDate>2014-10-31</EffectiveDate>
     <Categories>
      <Work_Location>Massachusetts</Work_Location>
     </Categories>
    </LaunchedEvent>
   </LaunchedEvents>
   <Note>0 events not launched</Note>
   <Note>Parsing started at Tuesday, November 11, 2014 11:27:22 AM and ended at Tuesday, November
11, 2014 11:27:22 AM</Note>
  </LaunchEventResults>
 </Data>
</EpsStringEx>
```

Element Syntax	Value	Description
LaunchEventResults	Parent node.	
Note	X event created	Successful event count. Present for Success and Failure.
	X event not created	Unsuccessful event count. Present for Success and Failure.
	Parsing started at...	Date and Time of method execution. Present for Success and Failure.
	Total duration:.x seconds (seconds were spent creating events)	Time to execute method. Present for Success and Failure.
EventError	Parent node	Exists only for an Event that fails

Element Syntax	Value	Description
ForWhom_*	unique identifier value of benefiting employee	ForWhom_ identifier to identify employee whose event failed
Name	Event name	Event name identifying event in error

Error	Parent node	The following Error format is returned for each error that occurs:
		`<Error>`
		`  <Type>See next table</Type>`
		`  <Message>See next table</Message>`
		`  <ErrorCode> See next table</ErrorCode>`
		`</Error>`

LaunchEvent Error Codes

<Type>	<Message>	<ErrorCode>	Description
Event Exists	*Named* event already exists for user: *ForWhom*	30000	Duplicate Event
Invalid Input	Failed to parse XML documents	30001	Caused by malformed XML
Invalid Input	Failed to add event to user as the sessionid executing the method is not an event coordinator	30002	Privilege error
Invalid Input	Unknown event name	30003	Undefined Event name
Invalid Input	Invalid category value name for *named category:value*	30004	The value in passed for <Type>Name</Type> did not match a valid category name

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Invalid category value code for *named* category:*value*	30005	The value in passed for <Type>Code</Type> did not match a valid category Code
Invalid Input	Invalid category value path for *named* category:*value*	30006	The value in passed for <Type>Path</Type> did not match a valid category Path
Invalid Input	Named category does not exist on the Event definition	30007	Launch data must map to field defined on the event definition.
Invalid Input	Named person does not exist on the Event definition	30008	Launch data must map to field defined on the event definition.
Invalid Input	The ForWhom_AuthParam used to identify a user is not used by any known user	30009	Invalid value external authorization parameter
Invalid Input	The ForWhom_LoginID used to identify a user is not used by any known user	30010	Invalid value employee LoginID
Invalid Input	The ForWhom_Guid used to identify a user is not used by any known user	30011	Invalid value User GUID
Invalid Input	The ForWhom_Email used to identify a user is not used by any known user	30012	Invalid value email address
Invalid Input	The e-mail address used to identify a user is used by more than one user	30013	Email address must be unique if it is to be used as the primary identifier

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	No user identification node could be found for the *nth* event node	30025	No valid ForWhom_* identifier was provided in the (specified) record
Invalid Input	Missing person (manager) name or Missing person (manager) value	30026	Person node included in record with no <Name> or <Value> node
Invalid Input	Missing EmployeeID Type for named person on the event	30056	Missing EmployeeIDType for a named person.
Invalid Input	Incorrect EmployeeIDType	31053	The EmployeeIDType {0} is unknown. Please use one of the following: LifeSuiteID, Employee_HRISID, Email, LoginID, GUID, or SSOAuthParam.
Invalid Input	Missing required information: *message text*	34000	A required event field specified in the message text has been omitted from the record.
Invalid Input	Invalid Category reference	36015	{0} cannot be used because it is not a leaf category value (i.e. It has child category values)

UpdateEvent

Usage

Used to update one or more attributes of an existing event for an active employee.

Parameters

Parameter	Required	Description	Type
strSecurityToken	✓	Valid Session ID for consuming service	simple string
updateEventXml	✓	XML record formatted following the rules outlined in "Field Descriptions of the xmlEventData (see page 164)".	simple string

The UpdateEvent method has the same requirements as updating an event through the Onboarding interface. The requirements are:

1. Only the fields to be updated need to be included in the <Event> record.
2. An employee must be active to have an event updated on their behalf.
3. *No new tasks* are generated for the employee at the time the event is successfully updated.
    - The task status will be adjusted based on any date modifications
    - The Task Summary page will correctly reflect the updated values of the Event in the Event Details section.
4. Task due dates based on an effected date will be adjusted.

Any correctly privileged user (event coordinators) can manually add task to existing events through the Onboarding user interface. (Click **Add a Task** on the Employee Task List page.)

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Data Node returns a node called <UpdateEventResults>

Examples

Example of "updateEventXml" input:

```
<?xml version="1.0" encoding="UTF-8"?>
<Events>
 <Event>
  <ForWhom_EmployeeID>
   <EmployeeIDType>LoginID</EmployeeIDType>
   <ID>Terri.Newa</ID>
  </ForWhom_EmployeeID>
  <Name>I-9 and E-Verify</Name>
  <Date>
   <Name>Start</Name>
   <Value>2014-11-04</Value>
  </Date>
    <Person>
        <Name>Manager</Name>
        <EmployeeID>
            <ID>888</ID>
            <EmployeeIDType>Employee_HRISID</EmployeeIDType>
        </EmployeeID>
    </Person>
 </Event>
</Events>
```

An example return string is as follows:

```
<EpsStringEx>
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <UpdateEventResults>
   <Note>1 events updated</Note>
   <UpdatedEvents>
    <UpdatedEvent>
     <ForWhomEmployee>
      <Guid>37c61606-af88-4f6e-a57e9c968a33f1d6</Guid>
      <Employee_HRISID>TNewa</Employee_HRISID>
      <SSOAuthParam>TNewa</SSOAuthParam>
      <LoginID>Terri.Newa</LoginID>
      <Email>tnewa@some.com</Email>
      <LifeSuiteID>50f64ae6-1563-4c22-9995-330cbba21641</LifeSuiteID>
      <FirstName>Terri</FirstName>
      <LastName>Newa</LastName>
```

```
      <MiddleName></MiddleName>
      <MiddleInitial></MiddleInitial>
     </ForWhomEmployee>
     <Name>I-9 and E-Verify</Name>
     <EventID>4</EventID>
     <EventStatus>Active</EventStatus>
     <CreationDate>2014-11-11T21:19:55</CreationDate>
     <EffectiveDate>2014-11-04</EffectiveDate>
     <Categories>
      <Work_Location>Massachusetts</Work_Location>
     </Categories>
    </UpdatedEvent>
   </UpdatedEvents>
   <Note>0 events not updated</Note>
   <Note>Parsing started at Wednesday, November 12, 2014 12:00:12 PM and ended at Wednesday,
November 12, 2014 12:00:12 PM</Note>
  </UpdateEventResults>
 </Data>
</EpsStringEx>
```

Note: If you are getting a subset of these fields in your results, you may have upgraded your site from a previous version. To obtain these results, you should remove the transformation applied to your site.

UpdateEvent Error Codes

The elements of the record returned in the data string follows:

Element Syntax	Value	Description
UpdateEventResults	Parent node.	
Note	X event created	Successful event count. Present for Success and Failure.

	X event not created	Unsuccessful event count. Present for Success and Failure.
	Parsing started at...	Date and Time of method execution. Present for Success and Failure.
	Total duration:.x seconds (seconds were spent creating events)	Time to execute method. Present for Success and Failure.
EventError	Parent node	Exists only for an Event that fails
ForWhom_*	unique identifier value of benefiting employee	ForWhom_ identifier to identify employee whose event failed
Name	Event name	Event name identifying event in error

Error	Parent node	The following Error format is returned for each error that occurs: <Error> <Type>See next table</Type> <Message>See next table</Message> <ErrorCode> See next table</ErrorCode> </Error>

Error codes

<Type> Value	<Message> Value	<ErrorCode> Value	Description
Invalid Input	Failed to parse XML documents	30001	Caused by malformed XML

<Type> Value	<Message> Value	<ErrorCode> Value	Description
Invalid Input	Failed to update event to user as the sessionid executing the method is not an event coordinator	30020	Privilege error
Invalid Input	Unknown event name	30003	Undefined Event name
Invalid Input	Invalid category value name for *named category:value*	30004	The value in passed for <Type>Name</Type> did not match a valid category name
Invalid Input	Invalid category value code for *named* category:*value*	30005	The value in passed for <Type>Code</Type> did not match a valid category Code
Invalid Input	Invalid category value path for *named* category:*value*	30006	The value in passed for <Type>Path</Type> did not match a valid category Path
Invalid Input	Named category does not exist on the Event definition	30007	Update data must map to field defined on the event definition.
Invalid Input	Named person does not exist on the Event definition	30008	Update data must map to field defined on the event definition.
Invalid Input	The ForWhom_AuthParam used to identify a user is not used by any known user	30009	Invalid value external auth parameters
Invalid Input	The ForWhom_LoginID used to identify a user is not used by any known user	30010	Invalid value employee LoginID
Invalid Input	The ForWhom_Guid used to identify a user is not used by any known user	30011	Invalid value User GUID

<Type> Value	<Message> Value	<ErrorCode> Value	Description
Invalid Input	The ForWhom_Email used to identify a user is not used by any known user	30012	Invalid value email address
Invalid Input	The e-mail address used to identify a user is used by more than one user	30013	Email address must be unique if it is to be used as the primary identifier
Invalid Input	No user identification node could be found for the *nth* event node	30025	No valid ForWhom_* identifier was provided in the (specified) record
Invalid Input	Missing person (manager) name or Missing person (manager) value	30026	Person node included in record with no <Name> or <Value> node
Invalid Input	Only categories can be modified on completed events.	30030	Event updates with a status of Complete are limited to category values.
Invalid Input	Missing EmployeeID Type for named person on the event	30056	Missing EmployeeIDType for a named person.
Invalid Input	ForWhom_x identifier node cannot be blank	31044	The ForWhom_x identifier specified in the message text is blank in the record.
Invalid Input	Incorrect EmployeeIDType	31053	The EmployeeIDType {0} is unknown. Please use one of the following: LifeSuiteID, Employee_HRISID, Email, LoginID, GUID, or SSOAuthParam.

<Type> Value	<Message> Value	<ErrorCode> Value	Description
Invalid Input	Missing required information: *message text*	34000	A required event field specified in the message text has been omitted from the record.
Invalid Input	Invalid Category reference	36015	{0} cannot be used because it is not a leaf category value (i.e. It has child category values)

Task APIs

Task APIs

These methods are available to interface with tasks from existing employee events:

- Add a new task for an existing event for an employee
- Delete an existing task from an existing event
- Get the existing tasks for an existing event for an employee
- Complete an existing active task

The "Task Definition ID" field defined in the task definition is the unique identifier of the task. The Task Definition ID is optional in the task definition but must exist to add, delete, or complete a task through the API.

Input specifications for GetTasks, CompleteTaskEx, and AddTaskEx2 Methods

GetTasks and CompleteTaskEx methods use similar arguments:

1. strSecurityToken - A Session ID.
    - The Onboarding user executing the methods must be a member of the Event Coordinator team for each event included in the strEventName parameter.
2. strXML - GetTasks, CompleteTaskEx, AddTaskEx2 accept an XML document to specify the input filters for the method.

Input specifications for Delete Task method

Note: DeleteTask is an older method that follows a different format. AddTask also exists for backward compatibility. If you are creating an integration or updating your integration you should use AddTaskEx2. Delete an existing task (DeleteTask) and Add a new task (AddTask) use similar arguments:

1. strSecurityToken - A Session ID.
    - The Onboarding user executing the methods must be a member of the Event Coordinator team for each event included in the strEventName parameter OR
    - The Onboarding user executing the methods must be a member of a team with ViewAllTasks permission
2. strForWhomUser - Any of the following employee profile attributes are supported: the employee Login ID, a unique email address, or the profile authparam. The strForWhomUser parameter also accepts the program generated userguid.
3. strEventName - The event name (in the default language) or event code as it is displayed on the Administration>Localization page.
4. strTaskDefinitionStringID - A unique identifier for the task as defined in the Task Definition ID field on the Edit Task Definition page

    After the Task Definition ID is entered and saved, it is visible on the Manage Tasks list. In the example below, the Task Definition ID is "OB_I9_1"

5. strSendEmails - True to send notification the task has been added, false to skip sending notifications.
6. strDontDeleteIfFormExists - Applies only to the DeleteTasks method. If a form is associated with the specified task (strTaskDefinitionStringID), a value of True prevents the task from being deleted. A value of False allows a task with an associated form to be deleted.

Output specifications

Each method returns an XML document of the following structure:

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

An XML record describing either the successful task data or a detail of the reason for failure.
The Data record xml is returned as CDATA tag.
If one or more parameters are in error, the ErrorNum returns a value not equal to 1. The XML in the Data string will describe the errors. The error text in the data string and error codes are described in the "Method Returns" section of each method description.

GetTasks

Usage

GetTasks returns task data for all tasks that meet the input criteria.
Filter input options include criteria to retrieve tasks by:

- "ForWhom" employee identifiers (the employee benefiting from the task) information such as employee ids, hire date range, key property value
- event information such as event definition code, event category values, event named people, and event effective date range
- task information such as task status, date ranges for that status, task definition IDs, or the task synchronization token (a database marker of when a given task was last touched).

The method returns task data including event name, "ForWhom" employee identifiers, task status, task name and definition id, task assignees, (actual), task activation date, modeled activation date, task completion date, who completed the task, task due date, and (if a form is associated with the task) the form type and object id.

If multiple events were launched for the employee that meet the selection criteria, the method returns the task data for the most recently launched event. The event output in 2015.1 and higher includes the event id and the task id. Event id and Task id are unique to each instance of the event / task.

- Event id is returned from LaunchEvent, UpdateEvent, AddTaskEx2 methods
- Task id is returned from AddTaskEx2

The maximum number of tasks returned in each call is controlled by the "GetTasks Max Records Returned" setting on the APIs tab on the Settings page of the application.. A node called "synchronization token" is provided to page through the next set of tasks that meet the input criteria.

- The synchronization token is an optional input parameter.
- The synchronization token is included in the output.

On the first call, the synchronization token (which defaults to null), is optionally passed as null. If your result set exceeds the configured number of records, you should call the method again passing the returned synchronization token. Subsequent calls will return the next records.

The tasks returned follow many of the same rules as the tasks available on the dashboard. Availability is configured by the variable (in the Onboarding Administration User interface) on the event definition "Days available on dashboard after completion"

Parameters

Parameter	Required	Description	Type
strSecurityToken	✓	Valid Session ID for consuming service	simple string
xmlEventData	✓	XML record formatted following the rules outlined in below	simple string

The valid session user must have permissions to view the tasks being requested. Permissions to view a task are granted to users on a team with View All Tasks privilege

Nodes

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	✓	XML Document header	
<GetTasksInput>	✓	Parent node	
<ForWhomEmployeeFilter>		Parent node	

Node	Required	Description	Notes
<EmployeeIDFilter>			
<EmployeeIDType></EmployeeIDType>		ID type	Acceptable values are: • Employee_HRISID • LoginID • SSOAuthParam • Email • Guid • LifeSuiteID See "Employee ID Types" (see page 106) for descriptions. example:<EmployeeIDType>LoginID</EmployeeIDType>
<ID></ID>		ID value	Specific to unique identifier for each user. This value is coupled to the Employee ID Type specified. Example: <ID>Tammy.Jones</ID>
<HireDateRange>		Parent node.	ForWhom employee Hire Date as entered on the Employee Profile detail section
<BeginDate></BeginDate>		YYYY-MM-DD format	Example: <BeginDate>2010-08-28</BeginDate>
<EndDate></EndDate>		YYYY-MM-DD format	Example: <EndDate>2010-09-28</EndDate>
</HireDateRange>		End parent node	
</EmployeeIDFilter>		End parent node	
</ForWhomEmployeeFilter>		End parent node	
<EventFilter>		Parent node	
<EventCode></EventCode>			Code from event definition to filter on.

Node	Required	Description	Notes

<EventStatus></EventStatus>			This node may appear more that once. Possible values are "In Progress", "Complete", and "Cancelled". Any combination of these 3 values may be used. When EventStatus is not used, the results will include tasks for "In Progress" and "Complete" events.
			`<EventStatus>In Progress</EventStatus>`
			`<EventStatus>Cancelled</EventStatus>`
			`<EventStatus>Complete</EventStatus>`

<CategoryValues>		Parent node	Event categories to filter on
<CategoryValue>		Parent node	1-many
<CategoryCode></CategoryCode>			
<CategoryValueCode></CategoryValueCode>			
</CategoryValue>		End parent node	
</CategoryValues>		End parent node	
<Person>		Parent node	Event people to filter, 0-many
<PersonCode></PersonCode>			Which event person
<EmployeeID></EmployeeID>		Parent node	Same as <EmployeeIDFilter> above
</Person>		End parent node	
<EffectiveDateRange>		Parent node	Event Effective Date range to filter results by, 0-1

Node	Required	Description	Notes
<BeginDate></BeginDate>		YYYY-MM-DD format	Example: <BeginDate>2010-08-28</BeginDate>
<EndDate></EndDate>		YYYY-MM-DD format	Example: <EndDate>2010-09-28</EndDate>
</EffectiveDateRange>		End parent node	
</EventFilter>		End parent node	
<TaskStatus></TaskStatus>			Valid values: • Complete • Incomplete And optionally for more granular types of Incomplete tasks: • Inactive • Expired • Active • Warning • Overdue Example: `<TaskStatus>Inactive</TaskStatus>` Using the "Active" filter will return tasks in the Active, Warning and Overdue states.
<TaskStatusDateRange>		Parent node, Occurs 1	Date range to use in filtering on this task status
<BeginDate></BeginDate>		YYYY-MM-DD format	Example: <BeginDate>2010-08-28</BeginDate>
<EndDate></EndDate>		YYYY-MM-DD format	Example: <EndDate>2010-09-28</EndDate>
</TaskStatusDateRange>		End parent node	
<TaskDefinitionIDFilter>		Parent node	

Node	Required	Description	Notes
<TaskDefinitionID> </TaskDefinitionID>		Occurs Unbounded	Simple string matching the task definition id defined in the Onboarding User Interface
</TaskDefinitionIDFilter>		Close parent node	
<SynchronizationToken></SynchronizationToken>		Default is null	Synchronization token marker for where to begin the results set. Use only when making repeat calls to page through a results set. A valid synchronization token (generated by Onboarding and obtained by a previous call to this method) returns the next page of results. The synchronization token is a simple string and should be stored as a string. Result set size is specified by the Onboarding setting "GetTasks Max Records Returned" <SynchronizationToken />indicates a null value.
<DaysToLinger on="true|false" />		Default is "true"	Controls whether the Days To Linger setting will be applied to the result query for completed tasks (RedCarpet 2019.1)
</GetTasksInput>	✓	close parent node	

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Data Node returns a node called <GetTasksResults>

GetTasks Error messages

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Invalid category alias Code: 'NonExist'	30042	
Invalid Input	Invalid event status: 'zzz'	30062	
Invalid Input	Invalid category value code for *X* category: *Y value*	30005	
Invalid Input	Unknown task definition string ID: NonExistantTask	35002	
Invalid Input	Key Property: Invalid category alias Code: *X*	39002	
Invalid Input	Key Property: Invalid category value code for NonExistCatValue category: *X*	39010	
Invalid Input	Key Property: Invalid person Code: *X*	39004	
Invalid Input	The LoginID (NonExistantUser) used to identify a user is not used by any known user	39011	
Invalid Input	Invalid input XML: *description of invalid element*	42000	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The synchronization token cannot be decrypted and appears to have been tampered with.	42002	

Examples

Example of GetTasks strXML to return a list of tasks for employees hired within a specific date range:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetTasksInput>
<ForWhomEmployeeFilter>
<HireDateRange>
    <BeginDate>2010-09-01</BeginDate>
    <EndDate>2010-09-28</EndDate>
</HireDateRange>
</ForWhomEmployeeFilter>
<TaskStatus>Active</TaskStatus>
</GetTasksInput>
```

Example of GetTasks strXML to return a list of expired tasks for an employee by SSO authentication parameter within a date range:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetTasksInput>
 <ForWhomEmployeeFilter>
  <EmployeeIDFilter>
   <EmployeeIDType>SSOAuthParam</EmployeeIDType>
   <ID>Jane.Smith</ID>
  </EmployeeIDFilter>
 </ForWhomEmployeeFilter>
 <TaskStatus>Expired</TaskStatus>
 <TaskStatusDateRange>
  <BeginDate>2010-09-01</BeginDate>
  <EndDate>2010-09-28</EndDate>
 </TaskStatusDateRange>
</GetTasksInput>
```

Example Output

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
<ErrorString>No Error</ErrorString>
<ErrorNum>1</ErrorNum>
<Data><![CDATA[
 <Tasks xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
```

```
  <Task>
   <TaskID>125</TaskID>
   <Event>
    <EventID>15</EventID>
    <EventName>I-9 &amp;amp; E-Verify</EventName>
    <EventCode>I9_eVerify_Code</EventCode>
    <EventStatus>Complete</EventStatus>
    <ForWhomEmployee>
     <Guid>7213ed6f-b057-4a8d-a9b69878301dc391</Guid>
     <Employee_HRISID>nnewhire</Employee_HRISID>
     <SSOAuthParam>nnewhire</SSOAuthParam>
     <LoginID>Nancy.NewHire</LoginID>
     <Email>Nancy.NewHire@silkroad.com</Email>
     <LifeSuiteID>c3df2041-22e6-4b04-91a0-890ea9994d28</LifeSuiteID>
    </ForWhomEmployee>
   </Event>
   <TaskStatus>Complete</TaskStatus>
   <Name>Complete Form I-9 Section 1</Name>
   <TaskDefinitionID>I9EV_FormI9_Sec1</TaskDefinitionID>
   <AssigneeTeam xsi:nil="true" />
   <AssigneeEmployee>
    <Guid>7213ed6f-b057-4a8d-a9b69878301dc391</Guid>
    <Employee_HRISID>nnewhire</Employee_HRISID>
    <SSOAuthParam>nnewhire</SSOAuthParam>
    <LoginID>Nancy.NewHire</LoginID>
    <Email>Nancy.NewHire@silkroad.com</Email>
    <LifeSuiteID>c3df2041-22e6-4b04-91a0-890ea9994d28</LifeSuiteID>
   </AssigneeEmployee>
   <ActivationDateTime>2014-11-05T13:29:28.547</ActivationDateTime>
   <ModeledActivationDate>2014-11-05</ModeledActivationDate>
   <CompletionDateTime>2014-11-05T13:32:01.65</CompletionDateTime>
   <CompletedByEmployee>
    <Guid>7213ed6f-b057-4a8d-a9b69878301dc391</Guid>
    <Employee_HRISID>nnewhire</Employee_HRISID>
    <SSOAuthParam>nnewhire</SSOAuthParam>
    <LoginID>Nancy.NewHire</LoginID>
    <Email>Nancy.NewHire@silkroad.com</Email>
    <LifeSuiteID>c3df2041-22e6-4b04-91a0-890ea9994d28</LifeSuiteID>
   </CompletedByEmployee>
   <DueDate>2014-11-04</DueDate>
   <Form>
    <ObjectId>13064</ObjectId>
    <FormType>Form_I_9_2009</FormType>
   </Form>
   <SynchronizationToken>SSg9M4zc12PCX35M8d+Puj8Sc6ThDcHv+lxLLrqTfLA=</ SynchronizationToken>
  </Task>
 </Tasks>]]>
</Data>
</EpsStringEx>
```

If an employee is returned all the identifiers are returned for the employee. For example:

```
<ForWhomEmployee>
 <Guid>54c37d74-587e-453a-ae3b97d0981cc9eb</Guid>
 <Employee_HRISID>CBrown100</Employee_HRISID>
 <SSOAuthParam>100100</SSOAuthParam>
 <LoginID>Charlie.Brown</LoginID>
 <Email>charlie.brown@silkroadtech.com</Email>
 <LifeSuiteID xsi:nil="true" />
</ForWhomEmployee>

<AssigneeEmployee>
 <Guid>a0b90424-ad07-41d7-bd97b8ab6f59916b</Guid>
 <Employee_HRISID>100</Employee_HRISID>
 <SSOAuthParam xsi:nil="true" />
 <LoginID>Amy.HrCoord</LoginID>
 <Email>Amy.HrCoord@mySRT.com</Email>
 <LifeSuiteID />
</AssigneeEmployee>

<CompletedByEmployee>
 <Guid>a0b90424-ad07-41d7-bd97b8ab6f59916b</Guid>
 <Employee_HRISID>100</Employee_HRISID>
 <SSOAuthParam xsi:nil="{*}true{*}" />
 <LoginID>Amy.HrCoord</LoginID>
 <Email>Amy.HrCoord@mySRT.com</Email>
 <LifeSuiteID />
</CompletedByEmployee>
```

If no employee is applicable such as the CompletedByEmployee on tasks not complete, the value is <CompletedByEmployee xsi:nil="true" />

The SynchronizationToken is used solely for paging through the results set. Use the token to tasks for each call that returns 500 tasks. The subsequent call should have the same input parameters with the exception of the new token value. You should not modify the value of the synchronization token.

Example return with errors:

```
<EpsStringEx>
 <ErrorString>Invalid input XML: The 'TaskStatus' element is invalid - The value 'Incomp' is
invalid according to its datatype 'String' - The Enumeration constraint failed.</ErrorString>
<ErrorNum>0</ErrorNum>
<Data><![CDATA[
 <GetTasksResults>
   <Error>
    <Type>Invalid Input</Type>
    <Message>Invalid input XML: The 'TaskStatus' element is invalid - The value 'Incomp' is
invalid according to its datatype 'String' - The Enumeration constraint failed.</Message>
    <ErrorCode>42000</ErrorCode>
  </Error>
 </GetTasksResults>
]]></Data>
</EpsStringEx>
```

AddTaskEx2

AddTaskEx2 and AddTaskEx methods

AddTaskEx2 is the most recent and flexible method of the Add Task options. AddTask and AddTaskEx exist for backward compatibility.

Usage

Either method can be used to programmatically add a dependent task on an existing event for an existing employee. Both methods provide the option to add the task as either an unassociated task or a dependent task. Both methods provide the option to send email.

If the added task is a dependent task in the workflow, the parent task is currently an incomplete task, and the parent task is *NOT* defined as "Always Reopen Dependent Tasks", you must use the AddTaskEx2 or AddTaskEx methods instead of the AddTask method.

AddTaskEx2 (and AddTask and AddTaskEx) methods allow adding dependent tasks if the parent task is defined as "Always Reopen Dependent Tasks".

AddTaskEx2 is the latest and most flexible version of the methods. If you have not already coded to AddTask or AddTaskEx you should use AddTaskEx2. AddTaskEx2 accepts an XML string as an input parameter allowing more flexibility for future changes to the method.

Code examples

The following is an example of the strXml input string for AddTaskEx2

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <AddTaskEx2Input>
3  <Event>
4  <EventCode>I9_eVerify_Code</EventCode>
5  <ForWhomEmployeeID>
6  <EmployeeIDType>Employee_HRISID</EmployeeIDType>
7  <ID>Ann.New</ID>
8  </ForWhomEmployeeID>
9  </Event>
```

```
<TaskDefinitionID>VI9F</TaskDefinitionID>
<SendEmails>true</SendEmails>
<AddDependant>false</AddDependant>
</AddTaskEx2Input>
```

Note:

> **Note:**
>
> See XML Schema Files for available XSDs for your development effort.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

AddTaskEx2 strXML nodes

Node	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	XML Document header	Required
<AddTaskEx2Input>	Parent node	Required
<Event>	Parent node	Required
<EventCode>	Event Code as defined on the Administration -> Localization page. Uniquely identifies the Event Definition.	Required
<ForWhomEmployeeID>	Parent node	Required

<EmployeeIDType></ <EmployeeIDType>	Type of ID value being specified in <ID></ID> node Occurs once	Valid Values: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteID Example: <EmployeeIDType>SSOAuthParam</ EmployeeIDType>
<ID></ID>	Unique identifier of ForWhom employee Occurs unbounded	Example: <ID>jane.smith</ID>
</ForWhomEmployeeID>	end parent node	
<TaskDefinitionID> </TaskDefinitionID>	Task Definition ID as defined on the Edit Task Definition page of the user interface. This string is unique to the task definition.	Example: <TaskDefinitionID>JY978</TaskDefinitionID>
<SendEmails></ SendEmails>	Boolean values true, false, or 1 • true to send a configured notification a task has been added. • false to not send a configured notification a task has been added. • 1 to send a configured notification a task has been added.	Example: <SendEmails>true</SendEmails>
<AddDependant></ AddDependant>	See additional information below. Possible values are true or false. Pass true to add the task as an inactive task to be activated after the parent is completed Pass false to disregard its parent. Passing false will add the task as an activate task unassociated task.	Example: <AddDependant>true</AddDependant>

Method returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

A record will be returned with descriptive XML

Nodes in AddTaskEx2 Data results

Node	Description
<AddTaskEx2Results>	
<Task>	
<TaskID></TaskID>	Unique task id generated by Onboarding for this employee / event/ task instance
<Event>	parent node
<EventID></EventID>	Unique event id generated by Onboarding for this employee / event instance when the event was launched.
<EventName></EventName>	Name of event as specified on the Administration-> Manage Events page of the user interface

<EventCode></EventCode>	Name of event as specified on the Administration -> Localization page of the user interface
<ForWhomEmployee>	Parent node identifying information on the ForWhom the task is added. This was specifiied by the method call
<Guid></Guid>	Unique employee id generated by Onboarding on employee creation
<Employee_HRISID> </Employee_HRISID>	Optional Unique employee id entered by employer
<SSOAuthParam> </SSOAuthParam>	Optional employee id entered by employer for Single Sign On (authentication)
<LoginID></LoginID>	Required unique employee id entered by employer
<Email>/Email>	Optional (by configuration)
<LifeSuiteID></LifeSuiteID>	Unique employee id generated by the SilkRoad LifeSuite reserved for future use.
</ForWhomEmployee>	end parent node
</Event>	end parent node
<TaskStatus></TaskStatus>	Status of task on addition. Possible values are: Complete Inactive Active Warning Overdue
<Name>/Name>	Name of task added
<TaskDefinitionID> </TaskDefinitionID>	Task Definition ID as defined in the Administration -> Manage Events -> Manage Tasks -
<AssigneeTeam />	Populated if the added task is a team task.
<AssigneeEmployee>	Parent node.

<Guid> </Guid>	Identifier of the assignee if the added task is assigned to a person.
<Employee_HRISID> </Employee_HRISID>	Identifier of the assignee if the added task is assigned to a person.
<SSOAuthParam> </SSOAuthParam>	Identifier of the assignee if the added task is assigned to a person.
<LoginID> </LoginID>	Identifier of the assignee if the added task is assigned to a person.
<Email> </Email>	Identifier of the assignee if the added task is assigned to a person.
<LifeSuiteID></LifeSuiteID>	Reserved for future use.
</AssigneeEmployee>	end parent node
<ModeledActivationDate></ ModeledActivationDate>	Activation date based on the rules defined in the task definition.
<ActivationDateTime> </ActivationDateTime>	Actual activation date time. This will be nill if the task just added is not yet activated.
<CompletionDateTime />	Completion date time. This will be nill if the task just added is not yet completed.
<CompletedByEmployee />	Completed set of employee ids (same format as AssigneeEmployee node. This will be nill if the task just added is not yet completed.
<DueDate></DueDate>	Due date based on the rules defined in the task definition
</Task>	end task node
</AddTaskEx2Results>	

```
1  <EpsStringEx>
```

```
<ErrorString>No Error</ErrorString>
<ErrorNum>1</ErrorNum>
<Data>
<AddTaskEx2Results xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Task>
<TaskID>26</TaskID>
<Event>
<EventID>2</EventID>
<EventName>I-9 & E-Verify</EventName>
<EventCode>I9_eVerify_Code</EventCode>
<ForWhomEmployee>
<Guid>363c6ab2-1564-469a-9139a0a9f0333f59</Guid>
<Employee_HRISID>Ann.New</Employee_HRISID>
<SSOAuthParam>Ann.New</SSOAuthParam>
<LoginID>Ann.New</LoginID>
<Email>ann.new@some.com</Email>
<LifeSuiteID>2cd300aa-b045-4735-963b-289720703e49</LifeSuiteID>
</ForWhomEmployee>
</Event>
<TaskStatus>Overdue</TaskStatus>
<Name>View I-9 Form</Name>
<TaskDefinitionID>VI9F</TaskDefinitionID>
<AssigneeTeam xsi:nil="true" />
<AssigneeEmployee>
<Guid>3482513d-954e-4f68-814d84e139e449a7</Guid>
<Employee_HRISID>AHRCoord</Employee_HRISID>
<SSOAuthParam>AHRCoord</SSOAuthParam>
<LoginID>Amy.HRCoord</LoginID>
<Email>jpostle@sikroad.com</Email>
<LifeSuiteID>f0ca053f-c145-44b2-ad10-f191ad03e306</LifeSuiteID>
</AssigneeEmployee>
<ModeledActivationDate>2014-11-11</ModeledActivationDate>
<ActivationDateTime>2014-11-12T15:27:57.753</ActivationDateTime>
<CompletionDateTime xsi:nil="true" />
<CompletedByEmployee xsi:nil="true" />
<DueDate>2014-11-11</DueDate>
</Task>
</AddTaskEx2Results>
</Data>
</EpsStringEx>
```

AddTaskEx

AddTaskEx2 and AddTaskEx methods

AddTaskEx2 is the most recent and flexible method of the Add Task options. AddTask and AddTaskEx exist for backward compatibility.

Usage

Either method can be used to programmatically add a dependent task on an existing event for an existing employee. Both methods provide the option to add the task as either an unassociated task or a dependent task. Both methods provide the option to send email.

If the added task is a dependent task in the workflow, and the parent task is currently an incomplete task, and the parent task is *NOT* defined as "Always Reopen Dependent Tasks", you must use the AddTaskEx2 or AddTaskEx methods instead of the AddTask method.

AddTaskEx2 (and AddTask and AddTaskEx) methods all allow adding dependent tasks if the parent task is defined as "Always Reopen Dependent Tasks".

AddTaskEx2 is the latest and most flexible version of the methods. If you have not already coded to AddTask or AddTaskEx you should use AddTaskEx2. AddTaskEx2 accepts an XML string as an input parameter allowing more flexibility for future changes to the method.

AddTaskEx2 code examples

The following is an example of the strXml input string for AddTaskEx2

```
<?xml version="1.0" encoding="UTF-8"?>
<AddTaskEx2Input>
<Event>
<EventCode>I9_eVerify_Code</EventCode>
<ForWhomEmployeeID>

<EmployeeIDType>Employee_HRISID</EmployeeIDType>
<ID>Ann.New</ID>
</ForWhomEmployeeID>
</Event>
<TaskDefinitionID>VI9F</TaskDefinitionID>
<SendEmails>true</SendEmails>
<AddDependant>false</AddDependant>
</AddTaskEx2Input>
```

Note:

See XML Schema Files for available XSDs for your development effort.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

AddTaskEx2 strXML nodes

Node	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	XML Document header	Required
<AddTaskEx2Input>	Parent node	Required
<Event>	Parent node	Required
<EventCode>	Event Code as defined on the Administration -> Localization page. Uniquely identifies the Event Definition.	Required
<ForWhomEmployeeID>	Parent node	Required
<EmployeeIDType></ <EmployeeIDType>	Type of ID value being specified in <ID></ID> node Occurs once	Valid Values: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteID Example: <EmployeeIDType>SSOAuthParam</ EmployeeIDType>
<ID></ID>	Unique identifier of ForWhom employee Occurs unbounded	Example: <ID>jane.smith</ID>
</ForWhomEmployeeID>	end parent node	

<TaskDefinitionID> </TaskDefinitionID>	Task Definition ID as defined on the Edit Task Definition page of the user interface. This string is unique to the task definition.	Example: <TaskDefinitionID>JY978</TaskDefinitionID>
<SendEmails> </SendEmails>	Boolean values true, false, or 1 • true to send a configured notification a task has been added. • false to not send a configured notification a task has been added. • 1 to send a configured notification a task has been added.	Example: <SendEmails>true</SendEmails>
<AddDependant> </AddDependant>	See additional information below. Possible values are true or false. Pass true to add the task as an inactive task to be activated after the parent is completed. Pass false to disregard its parent. Passing false will add the task as an activate task unassociated task.	Example: <AddDependant>true</AddDependant>

Method returns

Error messages are shared between AddTask, AddTaskEx, and AddTaskEx2.

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

A record will be returned with descriptive XML

Nodes in AddTaskEx2 Data results:

Node	Description
<AddTaskEx2Results>	
<Task>	
<TaskID></TaskID>	Unique task id generated by Onboarding for this employee / event/ task instance
<Event>	parent node
<EventID></EventID>	Unique event id generated by Onboarding for this employee / event instance when the event was launched.
<EventName></EventName>	Name of event as specified on the Administration-> Manage Events page of the user interface
<EventCode></EventCode>	Name of event as specified on the Administration -> Localization page of the user interface
<ForWhomEmployee>	Parent node identifying information on the ForWhom the task is added. This was specifiied by the method call
<Guid></Guid>	Unique employee id generated by Onboarding on employee creation

<Employee_HRISID> </Employee_HRISID>	Optional Unique employee id entered by employer
<SSOAuthParam> </SSOAuthParam>	Optional employee id entered by employer for Single Sign On (authentication)
<LoginID></LoginID>	Required unique employee id entered by employer
<Email>/Email>	Optional (by configuration)
<LifeSuiteID></LifeSuiteID>	Unique employee id generated by the SilkRoad LifeSuite reserved for future use.
</ForWhomEmployee>	end parent node
</Event>	end parent node
<TaskStatus></TaskStatus>	Status of task on addition. Possible values are: Complete Inactive Active Warning Overdue
<Name>/Name>	Name of task added
<TaskDefinitionID> </TaskDefinitionID>	Task Definition ID as defined in the Administration -> Manage Events -> Manage Tasks -
<AssigneeTeam />	Populated if the added task is a team task.
<AssigneeEmployee>	Parent node.
<Guid> </Guid>	Identifier of the assignee if the added task is assigned to a person.
<Employee_HRISID> </Employee_HRISID>	Identifier of the assignee if the added task is assigned to a person.
<SSOAuthParam> </SSOAuthParam>	Identifier of the assignee if the added task is assigned to a person.

<LoginID> </LoginID>	Identifier of the assignee if the added task is assigned to a person.
<Email> </Email>	Identifier of the assignee if the added task is assigned to a person.
<LifeSuiteID></LifeSuiteID>	Reserved for future use.
</AssigneeEmployee>	end parent node
<ModeledActivationDate></ ModeledActivationDate>	Activation date based on the rules defined in the task definition.
<ActivationDateTime> </ActivationDateTime>	Actual activation date time. This will be nill if the task just added is not yet activated.
<CompletionDateTime />	Completion date time. This will be nill if the task just added is not yet completed.
<CompletedByEmployee />	Completed set of employee ids (same format as AssigneeEmployee node. This will be nill if the task just added is not yet completed.
<DueDate></DueDate>	Due date based on the rules defined in the task definition
</Task>	end task node
</AddTaskEx2Results>	

```
<EpsStringEx>
<ErrorString>No Error</ErrorString>
<ErrorNum>1</ErrorNum>
<Data>
<AddTaskEx2Results xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
<Task>
<TaskID>26</TaskID>
<Event>
<EventID>2</EventID>
<EventName>I-9 & E-Verify</EventName>
```

```
<EventCode>I9_eVerify_Code</EventCode>
<ForWhomEmployee>
<Guid>363c6ab2-1564-469a-9139a0a9f0333f59</Guid>
<Employee_HRISID>Ann.New</Employee_HRISID>
<SSOAuthParam>Ann.New</SSOAuthParam>
<LoginID>Ann.New</LoginID>
<Email>ann.new@some.com</Email>
<LifeSuiteID>2cd300aa-b045-4735-963b-289720703e49</LifeSuiteID>
</ForWhomEmployee>
</Event>
<TaskStatus>Overdue</TaskStatus>
<Name>View I-9 Form</Name>
<TaskDefinitionID>VI9F</TaskDefinitionID>
<AssigneeTeam xsi:nil="true" />
<AssigneeEmployee>
<Guid>3482513d-954e-4f68-814d84e139e449a7</Guid>
<Employee_HRISID>AHRCoord</Employee_HRISID>
<SSOAuthParam>AHRCoord</SSOAuthParam>
<LoginID>Amy.HRCoord</LoginID>
<Email>jpostle@sikroad.com</Email>
<LifeSuiteID>f0ca053f-c145-44b2-ad10-f191ad03e306</LifeSuiteID>
</AssigneeEmployee>
<ModeledActivationDate>2014-11-11</ModeledActivationDate>
<ActivationDateTime>2014-11-12T15:27:57.753</ActivationDateTime>
<CompletionDateTime xsi:nil="true" />
<CompletedByEmployee xsi:nil="true" />
<DueDate>2014-11-11</DueDate>
</Task>
</AddTaskEx2Results>
</Data>
</EpsStringEx>
```

AddTaskNoteEx

AddTaskNoteEx method

AddTaskNoteEx is the most recent and flexible method to add task notes programmatically to existing tasks. An option exists on each task definition to send a notification when a task note is added. Recipients for each note type are specified (configured) in the Onboarding administration UI (Manage Notifications). AddTaskNoteEx has an input parameter to specify if a notification will be sent (in addition to adding a task note visible in the task history and audit report).

AddTaskNoteEx code examples

The following is an example of the strXml input string for AddTaskNoteEx

```
<?xml version="1.0" encoding="UTF-8"?>
<AddTaskNoteExInput>
<Task>
<Event>
<EventCode>I9_eVerify_Code</EventCode>
<ForWhomEmployeeID>
```

```
<EmployeeIDType>Employee_HRISID</EmployeeIDType>
<ID>Ann.New</ID>
</ForWhomEmployeeID>
</Event>
<TaskDefinitionID>VI9ACQ</TaskDefinitionID>
</Task>
<SendEmails>true</SendEmails>
<NoteText>Review required after acquisition</NoteText>
<AnnotatedByEmployeeID>
<EmployeeIDType>Employee_HRISID</EmployeeIDType>
<ID>Linda.HRCoord</ID>
</AnnotatedByEmployeeID>
</AddTaskNoteExInput>
```

Note

See XML Schema Files for available XSDs for your development effort.

Nodes in AddTaskNoteEx strXML

Node	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	XML Document header	Required
<AddTaskNoteExInput>	Parent node	Required
<Event>	Parent node	Required
<EventCode>	Event Code as defined on the Administration -> Localization page. Uniquely identifies the Event Definition.	Required
<ForWhomEmployeeID>	Parent node	Required
<EmployeeIDType></ <EmployeeIDType>	Type of ID value being specified in <ID></ID> node Occurs once	Valid Values: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteID Example: <EmployeeIDType>SSOAuthParam</ EmployeeIDType>

<ID></ID>	Unique identifier of ForWhom employee Occurs unbounded	Example: <ID>jane.smith</ID>
</ForWhomEmployeeID>	end parent node	
<TaskDefinitionID> </TaskDefinitionID>	Task Definition ID as defined on the Edit Task Definition page of the user interface. This string is unique to the task definition.	Example: <TaskDefinitionID>JY978</ TaskDefinitionID>
<SendEmails>	Boolean values true or false	Example:
</SendEmails>	true to send a configured notification a task has been added. false to not send a configured notification a task has been added.	<SendEmails>true</SendEmails>
<NoteText></NoteText>	Note to be added to the task.	
<AnnotatedByEmployeeID>		
<EmployeeIDType>	Type of ID value being specified in <ID></ID> node Occurs once	Valid Values: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteID Example: <EmployeeIDType>SSOAuthParam</ EmployeeIDType>
<ID></ID>	Unique identifier of ForWhom employee	Example: <ID>jane.smith</ID>
</EmployeeIDType>		
</AnnotatedByEmployeeID>		
</AddTaskNoteExInput>		

Methods available for backward compatibility

Code examples

To add a task with a Task Definition ID of "Transfer_Task1" to an event called "Transfer" for the employee with the loginID of "Jackson.Smith":

```
strForWhomUser = "Jackson Smith"
eprise.epsstringex ret = service.AddTaskEx(sessionnum,strForWhomUser, "Transfer",
"Transfer_Task1", "true", "true");
if (ret.errornum != 1)
{
console.out.writeline("error while adding task" + " error:" + ret.data); return;
}
```

Parameters

strSecurityToken	Required. Valid Session ID for consuming service (see privilege specifications in the Input (see page 0) Specifications for GetTasks, CompleteTaskEx, and AddTaskEx2 (see page 0) Methods (see page 0)section).	simple string
strForWhomUser	Required. The value refers to the Onboarding employee for whom the task will be added. Any of the following employee profile attributes are supported: the employee Login id, a unique email address, or the profile authparam. The strForWhomUser parameter will also accept the program generated userguid	simple string
string strEventName	Required. The value refers to the event for which the task will be added. The event name (in the default language) or event code as it is displayed on the Administration -> Localization page.	simple string
strTaskDefinitionStringId	Required. A unique identifier for the task as defined in the "Task Definition ID" field on the Edit Task Definition page See Input (see page 0) Specifications for GetTasks, CompleteTaskEx, and AddTaskEx2 (see page 0) Methods (see page 0)for additional information.	simple string
strSendEmails	Required Possible values are: • true to send notification a task has been added • false to not send notification a task has been added	simple string

strAddAsDependentTask EvenIfCanBeIndependent	Required See additional information below. Possible values are true or false. • Pass true to add the task as an inactive task to be activated after the parent is completed • Pass false to disregard it's parent. Passing false will add the task as an activate task unassociated task	simple string

If multiple events matching the specified strEventName exist for the employee, the task will be added to the (single) "in progress" event. If no "in progress" event exists, the task will be added to the last completed event. The event will be set to "in progress" (similar to adding a task to a completed event in the user interface).
If the added task is a dependent task in the workflow, and the parent task is currently an incomplete task, and the parent task is *NOT* defined as "Always Reopen Dependent Tasks", the strAddAsDependentTaskEvenIfCanBeIndependent parameter signifies:

- True to add the task as an inactivate task to be activated after the parent is completed.
- False to add the task as an activate task unassociated task.

Method returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

ErrorNum
This variable indicates success or failure. Values are:

- (one) — method succeeded
- (zero) — method failed Data

A record will be returned with descriptive XML
A sample return is as follows:

```
<EpsStringEx>
<ErrorString>Specified user's event already has specified task</ ErrorString>
<ErrorNum>0</ErrorNum>
<Data>
<AddTaskEx2Results xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
<Error>
<Type>Error</Type>
Message>
<Message>Specified user's event already has specified task</
<ErrorCode>35005</ErrorCode>
</Error>
</AddTaskEx2Results>
</Data>
</EpsStringEx>
```

The elements of the record returned in the data string are as follows:

Element Syntax Value Description
Data Parent node.
AddTaskEx2Results Parent node
ErrorParent nodeExists only for a method that fails.

The following Error format is returned for each error that occurs:
<Error>
<Type>See next table</Type>
<Message>See next table</Message>
<ErrorCode> See next table</ErrorCode>
</Error>
The following message and error codes apply to AddTaskEx and AddTaskEx2:

		Value	
Error	Unknown user identifier: {0}	30047	Value passed in the strForWhomUser variable must be a Login ID, email address, valid userguid, or authparam

Error	The e-mail address used to identify a user is used by more than one user	30013	Onboarding does not enforce unique e- mail address by default. In order to use an email as a unique identifier, the email must be unique
Error	{0} event doesn't exist for user: {1}	30014	Specified Event must exist for employee specified in the strForWhomUser
Error	Unknown event name or code: xxx	30046	strEventName must contain a valid event name or code (as displayed on the Location page)
Error	Unknown task definition string ID: xxx	35002	strTaskDefinitionStringID must contain a valid Task Definition ID (as displayed on the Manage Tasks page)
Error	I9 tasks cannot be added	35003	I9 Tasks can not be programmatically added
Error	Specified event definition does not have specified task definition	35004	strTaskDefinitionStringID must contain a valid Task Definition ID for the specified event (as displayed on the Manage Tasks page)
Error	Specified user's event already has specified task	35005	An existing task can not be added.
Error	Insufficient privileges	34001	See Input Specifications for GetTasks, (see page 0) CompleteTaskEx, and AddTaskEx2 (see page 0) Methods (see page 0)for session id privilege criteria.

<Type> Value <Message>Value <ErrorCode>

Description

```
<EpsStringEx>
<ErrorString>Specified user's event already has specified task</ ErrorString>
<ErrorNum>0</ErrorNum>
<Data>
<AddTaskEx2Results xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
<Error>
<Type>Error</Type>
Message>
<Message>Specified user's event already has specified task</
<ErrorCode>35005</ErrorCode>
</Error>
```

```
</AddTaskEx2Results>
</Data>
</EpsStringEx>
The elements of the record returned in the data string are as follows:
Element Syntax Value Description
Data Parent node.
AddTaskEx2ResultsParent node ErrorParent nodeExists only for a method that fails.
The following Error format is returned for each error that occurs:
<Error>
<Type>See next table</Type>
<Message>See next table</Message>
<ErrorCode> See next table</ErrorCode>
</Error>
```

CompleteTaskEx

Usage

CompleteTaskEx completes a specified active task. The task is identified by task definition ID for a specific benefiter ("for whom employee"). The input parameters include who is completing the task, task notes and if a notification should be sent. The CompleteTaskEx method does not allow the caller to complete tasks if any of the following conditions are true:

- if a task is marked as complete with the (associated) form is complete
- if a task has a "viewform" instruction with a form associated.
- if a task is a parent task (in the task hierarchy) and has an optional form

The output is a success or failure status on completing the task.

Input

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

CompleteTasksEx strXML examples

The valid session user must have permission to complete the tasks being requested. Permissions to view a task are granted to users on a team with View All Tasks privilege.

```
<?xml version="1.0" encoding="utf-8"?>
<CompleteTaskExInput>
<Task>
<Event>
<EventCode>Event_7</EventCode>
<ForWhomEmployeeID>
<EmployeeIDType>LoginID</EmployeeIDType>
```

```
<ID>Charlie.Brown</ID>
</ForWhomEmployeeID>
</Event>
<TaskDefinitionID>WS_ProvideKeyCard</TaskDefinitionID>
</Task>
<SendEmails>false</SendEmails>
<CompletedByEmployeeID>
<EmployeeIDType>LoginID</EmployeeIDType>
<ID>Amy.Jackson</ID>
</CompletedByEmployeeID>
</CompleteTaskExInput>Example of CompleteTasksEx strXML to complete a task with a Task Definition ID of WS_ProvideKeyCard for Charlie.Brown, completed by Amy.Jackson.:
```

CompleteTaskEx strXML nodes

Node	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	XML Document header	Required
<CompleteTaskExInput>	Parent node	Required
<Task>	Parent node	Required
<Event>	Parent node	Required
<EventCode></EventCode>	Simple string. Unique identifier for each event. Event code is defined in the Onboarding User interface on the Localizations page.	Required
<ForWhomEmployeeID>	Parent node	Required
<EmployeeIDType></EmployeeIDType>	Type of ID value being specified in <ID></ID> node Occurs once	Required Valid Values: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteID Example: <EmployeeIDType>SSOAuthParam</EmployeeIDType>
<ID></ID>	Unique identifier of ForWhom employee Occurs once	Required Example: <ID>jane.smith</ID>

</Event>	Close Event Parent node	
<TaskDefinitionID> </TaskDefinitionID>	Unique identifier specified on task definition in the Onboarding user interface	Required Example: <TaskDefinitionID>WelcomeLetter_Bos ton </TaskDefinitionID>
</Event>	Close Event Parent node	
<TaskDefinitionID> </TaskDefinitionID>	Unique identifier specified on task definition in the Onboarding user interface	Required Example: <TaskDefinitionID>WelcomeLetter_Bos ton </TaskDefinitionID>
</Task>	Close Task Parent node	
<SendEmails><SendEmails/>	Indicates if a completed task notification should be sent	Optional. Valid Values: true false Example: <SendEmailsNoteText>false</ SendEmailsNoteText>
<NoteText></NoteText>	Simple string. Text for task note.	Optional
<CompletedByEmployeeID>	Parent node Identifier of who task is completed by	Optional If CompletedByEmployeeID is not specified, the task note will indicate the task was completed by the API user (who obtained the session-id)
<EmployeeIDType></ EmployeeIDType>	Type of ID value being specified in <ID></ ID> node Occurs once	Required Valid Values: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteID Example: <EmployeeIDType>SSOAuthParam< /EmployeeIDType>
<ID></ID>	Unique identifier of ForWhom employee Occurs once	Required Example: <ID>jane.smith</ID>
</CompletedByEmployeeID>	End parent node	
</CompleteTaskExInput>	End parent node	

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Node returned with descriptive XML containing

```
![CDATA[<CompleteTaskExResults></CompleteTaskExResults>]]]

The elements of the record returned in the data string are as follows:
```

Element Syntax	Value	Description
Data	Parent node.	
CompleteTaskExResults	Parent node	Wrapped in CDATA
Error	Parent node	This node is included only for a method that fails. The following Error format is returned for each error that occurs: <Error> <Type>See next table</Type> <Message>See next table</Message> <ErrorCode> See next table</ErrorCode> </Error>

The following message and error codes apply to CompleteTasksEx:

<Type> Value	<Message>Value	<ErrorCode> Value
Error	Unknown task definition string ID: *Y*	35002
Error	Unknown event name or code: *X*	30046
Invalid Input	No user could be found for the id:UserItem:LoginID:*X*	31060
Error	Unknown task definition string ID: *X*	35002
Validation Error	Not authorized to complete the task.	35019
Error	Specified user's event does not have a specified task	35006
Error	You must be on the form to complete this task	32053
Error	You must opt-out of the form to complete this task	35024

<Type> Value <Message>Value <ErrorCode>

Sample returns

Example of the output of a successful call:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
  <ErrorString>No Error</ErrorString>
  <ErrorNum>1</ErrorNum>
<Data>
<![CDATA[ <CompleteTaskExResults /> ]]>
</Data>
</EpsStringEx>
```

Example output with an error:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
  <ErrorString>Unable to complete task.</ErrorString>
  <ErrorNum>0</ErrorNum>
```

```
<Data>
  <![CDATA[ <CompleteTaskExResults><Error><Type>Error</Type><Message>Not authorized to complete
the task.</Message><ErrorCode>35019</ErrorCode></Error></CompleteTaskExResults>]]>
</Data>
</EpsStringEx>
```

DeleteTask Method

Usage

Used to programmatically delete a task from an existing event for an existing employee.

Code examples

To delete a task with a Task Definition ID of "Transfer_Task1 to an event called "Transfer" for the employee with the loginID of "Jackson.Smith" the example syntax is as follows:.

```
strForWhomUser = “Jackson Smith”
eprise.epsstringex ret = service.DeleteTask(sessionnum,strForWhomUser, “Transfer”,
“Transfer_Task1”, “true”);
if (ret.errornum != 1)
{
    console.out.writeline("error while adding task" + " error:" + ret.data);
    return;
    }
```

Parameters

strSecurityToken	Required. Valid Session ID for consuming service (see privilege specifications in the Input (see page 0) Specifications for GetTasks, CompleteTaskEx, and AddTaskEx2 (see page 0) Methods (see page 0)section).	simple string
strForWhomUser	Required. Any of the following employee profile attributes are supported: the employee Login id, a unique email address, or the profile authparam. The strForWhomUser parameter will also accept the program generated userguid	simple string
string strEventName	Required. The event name (in the default language) or event code as it is displayed on the Administration -> Localization page.	simple string

strTaskDefinitionStringId	Required. A unique identifier for the task as defined in the "Task Definition ID" field on the Edit Task Definition page See Input (see page 0) Specifications for GetTasks, CompleteTaskEx, and AddTaskEx2 (see page 0) Methods (see page 0)for additional information.	simple string
strSendEmails	Required Possible values are: • True - to send notification a task has been added. • False - to not send notification a task has been added.	simple string
strdontDeleteIfFormExists	Required Possible values are: • True - Task will not be deleted if the task has a form. • False - both the task and form will be deleted	simple string

If multiple events matching the specified strEventName exist for the employee, the task will be deleted from the (single) "in progress" event.

Method returns

A table of type EpsStringEx. The structure is:

```
EpsStringEx{

    String ErrorString;
    int ErrorNum;
    String Data;
    }
```

ErrorString
Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
Error codes are returned in the same format for the post and the method call.

ErrorNum
This variable indicates success or failure. Values are:

- (one) — method succeeded
- (zero) — method failed

Data

A record will be returned with descriptive XML
A sample return is as follows:

```
<EpsStringEx>
  <ErrorString>Error adding task.</ErrorString>
  <ErrorNum>0</ErrorNum>
  <Data>
  <DeleteTaskResults>
    <Error>
      <Type>Error</Type>
      <Message>Specified user's event does not have specified task</Message>
      <ErrorCode>35006</ErrorCode>
    </Error>
   </DeleteTaskResults>
  </Data>

</EpsStringEx>
```

The elements of the record returned in the data string are as follows:

Element Syntax	Value	Description
Data	Parent node.	
DeleteTaskResults	Parent node	
Error	Parent node	This node is included only for a method that fails. The following Error format is returned for each error that occurs: <Error> <Type>See next table</Type> <Message>See next table</Message> <ErrorCode> See next table</ErrorCode> </Error>

The following message and error codes apply to DeleteTask:

<Type> Value	<Message>Value	<ErrorCode>Value	Description

Error	Unknown user identifier: {0}	30047	Value passed in the strForWhomUser variable must be a Login ID, email address, valid userguid, or authparam
Error	The e-mail address used to identify a user is used by more than one user	30013	Onboarding does not enforce unique e-mail address by default. In order to use an email as a unique identifier, the email must be unique
Error	{0} event doesn't exist for user: {1}	30014	Specified Event must exist for employee specified in the strForWhomUser
Error	Unknown event name or code: xxx	30046	strEventName must contain a valid event name or code (as displayed on the Location page)
Error	Unknown task definition string ID: xxx	35002	strTaskDefinitionStringID must contain a valid Task Definition ID (as displayed on the Manage Tasks page)
Error	I9 tasks cannot be deleted	35000	I9 Tasks can not be programmatically deleted
Error	Specified event definition does not have specified task definition	35004	strTaskDefinitionStringID must contain a valid Task Definition ID for the specified event (as displayed on the Manage Tasks page)
Error	Specified user's event does not have specified task	35006	The task must exist to be deleted.
Error	Insufficient privileges	34001	See Input Specifications for GetTasks, (see page 0) CompleteTaskEx, and AddTaskEx2 (see page 0) Methods (see page 0)for session id privilege criteria.

AddTask Method

Usage

Used to programmatically add a task on an existing event for an existing employee.

Code examples

To add a task with a Task Definition ID of "Transfer_Task1" to an event called "Transfer" for the employee with the loginID of "Jackson.Smith" the example syntax is:

```
strForWhomUser = "Jackson Smith"
eprise.epsstringex ret = service.AddTask(sessionnum,strForWhomUser, "Transfer",
"Transfer_Task1", "true");
if (ret.errornum != 1)
{
    console.out.writeline("error while adding task" + " error:" + ret.data); return;
    }
```

Parameters

strSecurityToken	Required. Valid Session ID for consuming service (see privilege specifications in the Input (see page 0) Specifications for GetTasks, CompleteTaskEx, and AddTaskEx2 (see page 0) Methods (see page 0)section).	simple string
strForWhomUser	Required. The value refers to the Onboarding employee for whom the task will be added. Any of the following employee profile attributes are supported: the employee Login id, a unique email address, or the profile authparam. The strForWhomUser parameter will also accept the program generated userguid	simple string
string strEventName	Required. The value refers to the event for which the task will be added. The event name (in the default language) or event code as it is displayed on the Administration>Localization page.	simple string
strTaskDefinitionStringId	Required. A unique identifier for the task as defined in the "Task Definition ID" field on the Edit Task Definition page See Input (see page 0) Specifications for GetTasks, CompleteTaskEx, and AddTaskEx2 (see page 0) Methods (see page 0)for additional information.	simple string

strSendEmails	Required Possible values are: • True to send notification a task has been added. • False to not send notification a task has been added. • 1 to send notification a task has been added.	simple string

Method Returns

A sample return is as follows:

```
<EpsStringEx>
  <ErrorString>Error adding task.</ErrorString>
  <ErrorNum>0</ErrorNum>
  <Data>
  <AddTaskResults>
    <Error>
      <Type>Error</Type>
      <Message>Specified user's event already has specified task</Message>
      <ErrorCode>35005</ErrorCode>
    </Error>
  </AddTaskResults>
 </Data>
</EpsStringEx>
```

The elements of the record returned in the data string are as follows:

Element Syntax	Value	Description
Data	Parent node.	
AddTaskResults	Parent node	

Error	Parent node	Exists only for a method that fails. The following Error format is returned for each error that occurs: <Error> <Type>See next table</Type> <Message>See next table</Message> <ErrorCode> See next table</ErrorCode> </Error>

The following message and error codes apply to AddTask:

<Type> Value	**<Message>Value**	**<ErrorCode>Value**	**Description**
Error	Unknown user identifier: {0}	30047	Value passed in the strForWhomUser variable must be a Login ID, email address, valid userguid, or authparam
Error	The e-mail address used to identify a user is used by more than one user	30013	Onboarding does not enforce unique e-mail address by default. In order to use an email as a unique identifier, the email must be unique
Error	{0} event doesn't exist for user: {1}	30014	Specified Event must exist for employee specified in the strForWhomUser
Error	Unknown event name or code: xxx	30046	strEventName must contain a valid event name or code (as displayed on the Location page)
Error	Unknown task definition string ID: xxx	35002	strTaskDefinitionStringID must contain a valid Task Definition ID (as displayed on the Manage Tasks page)
Error	I9 tasks cannot be added	35003	I9 Tasks can not be programmatically added
Error	Specified event definition does not have specified task definition	35004	strTaskDefinitionStringID must contain a valid Task Definition ID for the specified event (as displayed on the Manage Tasks page)

Error	Specified user's event already has specified task	35005	An existing task can not be added.
Error	Insufficient privileges	34001	See Input Specifications for GetTasks, (see page 0) CompleteTaskEx, and AddTaskEx2 (see page 0) Methods (see page 0)for session id privilege criteria.

Reports API

Overview

Two APIs are available to execute the Onboarding Events report. GetEventReportEx (released with RedCarpet 2.9.0) and GetEventReport (deprecated with the release of GetEventReportEx).

For customers with a high volume of events SilkRoad does **not** recommend using the GetEventReport or GeEventReportEx methods to obtain employee user profile and task data information.
See GetUserIDList[3], GetUserProfileEx2[4] and GetTasks[5] methods for additional information.

GetEventReportEx

Usage

Used to generate XML output using the same parameters available in the Onboarding user interface.

The columns included in the report output are specified in the Onboarding Settings page.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

Example Structure of GetEventReportEx strXML

```
<?xml version="1.0" encoding="utf-8"?>
<GetEventReportExInput>
  <EventFilter>
   <EventName>Onboarding</EventName>
   <CategoryFilter>
     <Name>Location</Name>
     <Value>BOS</Value>
   </CategoryFilter>
  </EventFilter>
  <EventStatus>Complete</EventStatus>
  <DateRange type='HireDate'>
  <BeginDate>2013-01-01</BeginDate>
  <EndDate>2013-07-02</EndDate>
```

3 https://confluence.silkroadtech.com/display/RED/GetUserIDList
4 https://confluence.silkroadtech.com/display/RED/GetUserProfileEx2
5 https://confluence.silkroadtech.com/display/RED/GetTasks

```
14    </DateRange>
15  </GetEventReportExInput>
```

XML input nodes

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	☐	XML Document header	
<GetEventReportExInput>	☐	Parent node	
<EventFilter>		Parent node. Optional (minOccurs = 0, maxOccurs = 1)	
<EventName></EventName>	☐	Required for any Event Filter	<EventName>Onboarding</ EventName>
<CategoryFilter>		Optional	Exist only if a EventName is specified
<Name></Name>	☐	Required if <CategoryFilter> is specified. (minOccurs = 1, maxOccurs = 1)	Valid values include category name (valid values are those available in the category export) <Name>Location</Name>
<Value></Value>	☐	Required if <CategoryFilter> is specified. (minOccurs = 1, maxOccurs = 1)	Valid values include both full value path or value code <Value>LOC_BOS</Value> The code value must match a value in the Category Export.
</CategoryFilter>	☐	Required if <CategoryFilter> is specified.	
</EventFilter>	☐	Required if <EventFilter> is specified.	
<EventStatus></EventStatus>		Optional (minOccurs = 0, maxOccurs = 1)	Valid Values: InProgress, Complete, Cancelled Example: <EventStatus>Complete</ EventStatus>

Node	Required	Description	Notes
<DateRange [type='Effective' (type attribute optional)]>		Optional (minOccurs = 0, maxOccurs = 1)	Valid types: Effective(default), Activation, Create, Due, Completed, HireDate) Example: <DateRange type='HireDate'>
<BeginDate></BeginDate>		Optional. (minOccurs = 0, maxOccurs = 1)	If BeginDate is not specified it will default to yesterday. YYYY-MM-DD format Example: <BeginDate>2013-06-02</BeginDate>
<EndDate></EndDate>		Optional. (minOccurs = 0, maxOccurs = 1)	If EndDate is not specified it will default to today. YYYY-MM-DD format Example: <EndDate>2013-07-02</EndDate>
</DateRange>	☐	Required if <DateRange> is specified. close node	
</GetEventReportExInput>	☐	close node	

Return

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

- **1** (one) — method succeeded
- **0** (zero) — method failed

<Data>

An xml document with event data. Event data includes the fields defined in your configuration. By default the fields are: EventCreationDate, EffectiveDate, EventCompletionDate, EventStatus, EmployeeAuthParam, EmployeeHireDate, EmployeeTerminationDate, EmployeeEmail, EmployeeStatus, EmployeeFirstName, and EmployeeLastName.

Note on returned categories: If you configure category to be returned, and the categories were defined with special characters not accepted for XML node names, Onboarding replaces the special character with an _ (underbar). If the replacement results in a non-unique category name, a number is appended to the category name.

For customers with a high volume of events, SilkRoad does **not** recommend using the GetEventReport or GeEventReportEx methods to obtain employee user profile and task data information. See GetUserIDList (see page 106), GetUserProfileEx2 (see page 111) and GetTasks (see page 184) methods for additional information.

GetAuditReport

Usage

Generates a report of audit notes for a particular user. The XML format is the same as the Audit Report function in the Onboarding UI.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

Example 1: EmployeeIDFilter used to find audit notes where the employee is the subject of the note.

Example Structure of GetAuditReport strXML input.

```
<GetAuditReportInput>
<EmployeeIDFilter>
   <EmployeeIDType>LoginID</EmployeeIDType>
   <ID>john.smith</ID>
</EmployeeIDFilter>
<MaximumRows>1000</MaximumRows>
<StartRowIndex>0</StartRowIndex>
<BeginDate>2018-01-01</BeginDate>
<EndDate>2018-07-02</EndDate>
</GetAuditReportInput>
```

Example 2: EmployeeIDFilter is optional. (New for RC 2020.2.2.)

Example 2: EmployeeIDFilter is optional.

```
<GetAuditReportInput>
<MaximumRows>1000</MaximumRows>
<StartRowIndex>0</StartRowIndex>
<BeginDate>2018-01-01</BeginDate>
<EndDate>2018-07-02</EndDate>
</GetAuditReportInput>
```

Example #3: RelationshipToAudit may be used to search for audit notes generated by the employee. (The employee is the author of the note.) (New for RC 2020.2.2.)

Example #3: RelationshipToAudit may be used to search for audits authored by the employee.

```
<GetAuditReportInput>
<EmployeeIDFilter>
   <EmployeeIDType>LoginID</EmployeeIDType>
   <ID>john.smith</ID>
   <RelationshipToAudit>Author</RelationshipToAudit>
</EmployeeIDFilter>
<MaximumRows>1000</MaximumRows>
<StartRowIndex>0</StartRowIndex>
<BeginDate>2018-01-01</BeginDate>
<EndDate>2018-07-02</EndDate>
</GetAuditReportInput>
```

Example #4: Optional TeamFilter may be used to find audits generated by any employee on the specified team. (Any member of the team is the author of the note.) (New for RC 2020.2.2.)

Example #4: Optional TeamFilter may be used to find audits generated by any employee on the specified team.

```
<GetAuditReportInput>
<TeamFilter>
 <TeamName>I9_EC</TeamName>
</TeamFilter>
<MaximumRows>1000</MaximumRows>
<StartRowIndex>0</StartRowIndex>
<BeginDate>2018-01-01</BeginDate>
<EndDate>2018-07-02</EndDate>
</GetAuditReportInput>
```

XML Input Nodes

Node	Required	Description	Notes
GetAuditReportInput		Parent node	Optional since RC 2020.2.2.
EmployeeIDFilter	⚠️	Parent node of the Employee ID filter	At least one of EmployeeIDFilter, TeamFilter and MazimumRows must be included
EmployeeIDType		Enumeration	Allowed values are: Employee_HRISID, HRISID, Guid, SSOAuthParam, Email, LoginID, LifeSuiteID
ID		Unique Identifier for the user.	Value depends on EmployeeIDType
RelationshipToAudit		Allow searching for audits where the employee is the Author or the Subject.	Allowed values: Author Subject (this is the default).
TeamFilter	⚠️	Parent node	At least one of EmployeeIDFilter, TeamFilter and MazimumRows must be included
TeamName		Name of a team in RedCarpet.	Will find audits generated by any employee on the specified team.
MaximumRows	⚠️	Specifies the max number of records in the result set.	If the result set contains N+1 results, where N was the MaximumRows value, this indicates that more results are available. Use StartRowIndex to fetch more results. For performance reasons, this setting is limited to 10000 rows. At least one of EmployeeIDFilter, TeamFilter and MazimumRows must be included

Node	Required	Description	Notes
StartRowIndex		Used to fetch multiple pages of results. This is a 0-based offset from the beginning. A value of 5 will cause the first 5 results to be skipped.	NOTE: Results are always sorted by creation time, NEWEST records FIRST.
BeginDate		Filters results on Date. Only results after this date will be returned.	
EndDate		Filters results on Date. Only results before this date will be returned.	

Return

An xml document containing the audit records. Results are always sorted by creation time, NEWEST records FIRST.

Error messages

<Type>	<Message>	<ErrorCode>	Description
Error	Failed to parse XML document : Invalid input XML: The element '{0}' has incomplete content. List of possible elements expected: '{1}'.	30001	0 - Parent Element 1 - Missing Child Element
Error	The AuthParam used to identify a user is not used by any known user	30009	
Error	The LoginID used to identify a user is not used by any known user	30010	
Error	The Guid used to identify a user is not used by any known user	30011	
Error	The e-mail address used to identify a user is not used by any known user	30012	
Error	The Employee_HRISID {0} used to identify a user is not used by any known user	31054	
Error	The LifeSuiteId {0} used to identify a user is not used by any known user	31055	

<Type>	<Message>	<ErrorCode>	Description
Error	Invalid team value: {0}	31006	

Examples

Sample audit data:

Sample Audit Data

```
<GetAuditReportResults>
  <Audit_Report>
    <Audit_Note>
        <NoteID>3973</NoteID>
        <Date>2018-11-15T16:27:11</Date>
        <IpAddr>172.16.37.50</IpAddr>
        <Author>Abraham Lincoln</Author>
        <Item>Event:Onboarding Task: Verify Personal Information</Item>
        <Message>The task was completed.</Message>
        <Source>Onboarding Portal</Source>
    </Audit_Note>
  </Audit_Report>
</GetAuditReportResults>
```

Privileges for GetAuditReport

The following privileges allow access to different data from the Audit report in through the API.

- **Reports -** This privilege impacts both the API and user interface and provides access to all the reports in the user interface and data from GetAuditReport and GetEventReportEX (see page 227)API methods. A user with this privilege has access to all the data in the audit report.
- **Audit Report -** This privilege only impacts the API and provides access to GetAuditReport. A user with this privilege does **not** have access to changes to the User Profile.
- **Audit Report including Profile Information -** This privilege only impacts the API and provides access to GetAuditReport. A user with this privilege has access to all the data in the audit report.

Maintaining system information

CategoryUpload

Usage

The Category Upload API allows a import and edit of the category values. Used to programmatically create or modify the category value hierarchy using an XML.

This web method does not run an initial validation pass on the XML.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
categoryXml	☐	XML record formatted as described in Chapter 2 of the Category Upload Guide. (separate from this document)	string
uploadFileName		Not applicable. Expect value is "" (empty string).	simple string

Returns

The Data property of the StringEx returned by the web method will contain an XML string containing the results of the transaction, including number of Category Values affected and, in case of failure, a list of errors that caused the transaction to terminate.

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
 <ErrorNum>1</ErrorNum>
 <Data>
  <![CDATA[ XML document with results]]>
 </Data>
</EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

String containing XML record with error description

Examples

Code examples

To import new RedCarpet users from an XML record.

```
string scategoryXml = "<?xml version="1.0" encoding="windows-1252"?>"+ "<root>"+
" <Email></Email>"+
" <Version>1</Version>"+
" <Operation>Update</Operation>"+ " <Categories>"+
" <Category>"+
" <Name>Region</Name>"+ " <CategoryValue>"+
" <Name>All Regions</Name>"+ " <Code>ALL_REG</Code>"+
" <CategoryValue>"+
" <Name>United States</Name>"+ " <Code>US</Code>"+
" <CategoryValue>"+
" <Name>New England</Name>"+ " <Code>US_NE</Code>"+
" <CategoryValue>"+
" <Name>Massachusetts</Name>"+ " <Code>US_MA</Code>"+
" <CategoryValue>"+
" <Name>Bedford</Name>"+ " <Code>US_MA_BED</Code>"+ " </CategoryValue>"+
" <CategoryValue>"+
" <Name>Framingham</Name>"+ " <Code>US_MA_FRA</Code>"+ " </CategoryValue>"+
" </CategoryValue>"+ " </CategoryValue>"+ " </CategoryValue>"+ " </CategoryValue>"+ "
</Category>"+
" </Categories>"+ "</root>"
Eprise.EpsStringEx ret = service.CategoryUpload(sessionnum,scategoryXml,""); if
 (ret.ErrorNum != 1)
{
Console.Out.WriteLine("Error while trying import users" + " Error:" + ret.ErrorString);
return;
}
```

Example return ret.Data String:

```
<?xml version="1.0"?>
<UpdateCodesResults>
 <Note>0 Category Values updated</Note>
```

```
 <CategoryUploadError>
  <Name>Location</Name>
  <Error>
   <Message>Input structure doesn't match database structure for Location</ Message>
   <ErrorCode>36002</ErrorCode>
  </Error>
</CategoryUploadError>
</UpdateCodesResults>
```

Upload finished

Root nodes of the response XML will be named based on the Upload operation as follows:

- Import = ImportCategoryValuesResults
- Update = UpdateCategoryValuesResults
- UpdateCodes = UpdateCodesResults
- Replace = ReplaceCategoyValuesResults
- ReplaceAll = ReplaceAllResults

CategoryUpload Error Messages

<Type>	<Message>	<ErrorCode>	Description
	Failed to create restore point.	36000	
	'Code' node doesn't exist for <name of Category Value>	36001	
	Input structure doesn't match database structure for <Name of Category>	36002	
	Code is too long for {0}	36003	
	Malformed category upload XML	36004	
	Category Value with code '<Code>' doesn't exist.	36005	
	Code must be specified to retire Category Value '<Category Value name>'	36006	
	Invalid Portal Role: '<Eprise role>'	36007	

<Type>	<Message>	<ErrorCode>	Description
	'Update' and 'Replace' require all Category Value Codes to be populated and are not supported in your environment. Please use 'ReplaceAll' to bulk update your category values	36008	
	'<Category Name>': only one root Category Value is allowed per Category	36009	
	Internal Code '<Code>' already exists. Internal Codes must be unique if specified.	36010	
	A Category with the name '<Category Name>' already exists	36011	
	Category '<Category Name>' does not exist.	36012	

CategoryUpload Operations

Operation Usage

The <Operation> node in the XML determines the type of modification that will be applied to the category hierarchy.

If the <Operation> node is not provided, the upload defaults to a fresh import and output an error if category values already exist.

Each operation is described below.

Import

Usage

Default operation used for the initial import of category values

Example Syntax

The following example imports a new category called "Region".

```
<?xml version="1.0" encoding="windows-1252"?>
<root>
    <Email/>
    <Version>1</Version>
    <Operation>Import</Operation>
    <Categories>
        <Category>
            <Name>Region</Name>
            <CategoryValue>
```

```
                <Name>All Regions</Name>
                <Code>ALL_REG</Code>
                <CategoryValue>
                    <Name>United States</Name>
                    <Code>US</Code>
                    <CategoryValue>
                        <Name>Massachusetts</Name>
                        <Code>US_MA</Code>
                        <CategoryValue>
                            <Name>Bedford</Name>
                            <Code>US_MA_BED</Code>
                        </CategoryValue>
                        <CategoryValue>
                            <Name>Framingham</Name>
                            <Code>US_MA_FRA</Code>
                        </CategoryValue>
                    </CategoryValue>
                    <CategoryValue>
                        <Name>North Carolina</Name>
                        <Code/>
                        <CategoryValue>
                            <Name>Winston-Salem</Name>
                            <Code/>
                        </CategoryValue>
                        <CategoryValue>
                            <Name>Durham</Name>
                            <Code/>
                        </CategoryValue>
                    </CategoryValue>
                </CategoryValue>
            </CategoryValue>
        </Category>
    </Categories>
</root>
```

UpdateCodes

Code values may be blank or if populated; must be a unique value.

Usage

Used only for updating (or populating) the Code fields. When this operation is specified, the hierarchical structure of the XML must exactly match the hierarchical structure of the category values stored in the database.

This applies on a per-category basis. For example, the user may only want to update Location codes. Within a specified Category node, the XML structure must match that of the database for that Category.

If the structure does not match, the operation fails and no fields are updated.

Because of the need for an exact match on the hierarchical structure, this operation should be used with the Export feature. For example an administrator can use the Export (Update Codes) option to create a base record, update the Code nodes in the XML and resubmit it for upload.

The value of the Code node will override the existing Code value. If the Code node is empty, the Code for the specified category value will be blanked out.

UpdateCodes allows the updating of retired category values, as well as the active ones. The UpdateCode export XML will contain both active and retired category values. There is no distinction between retired and active category values in the XML.

Example syntax:

The following example updates the code values of the previous example to add code values for the last two categories (Winston Salem and Durham).

```
<?xml version="1.0" encoding="windows-1252"?>
<root>
    <Email/>
    <Version>1</Version>
    <Operation>UpdateCodes</Operation>
    <Categories>
        <Category>
            <Name>Region</Name>
            <CategoryValue>
                <Name>All Regions</Name>
                <Code>ALL_REG</Code>
                <CategoryValue>
                    <Name>United States</Name>
                    <Code>US</Code>
                    <CategoryValue>
                        <Name>Massachusetts</Name>
                        <Code>US_MA</Code>
                        <CategoryValue>
                            <Name>Bedford</Name>
                            <Code>US_MA_BED</Code>
                        </CategoryValue>
                        <CategoryValue>
                            <Name>Framingham</Name>
                            <Code>US_MA_FRA</Code>
                        </CategoryValue>
                    </CategoryValue>
                    <CategoryValue>
                        <Name>North Carolina</Name>
                        <Code>US_NC</Code>
                        <CategoryValue>
                            <Name>Winston-Salem</Name>
                            <Code>US_NC_WS</Code>
                        </CategoryValue>
                        <CategoryValue>
                            <Name>Durham</Name>
                            <Code>US_NC_DUR</Code>
                        </CategoryValue>
                    </CategoryValue>
                </CategoryValue>
            </CategoryValue>
```

```
        </Category>
    </Categories>
</root>
```

Update

A prerequisite of using the Update operation is **all** code values must be populated (in the database)

> **Note:** Use the UpdateCodes operation to populate all code values before executing an Update operation. New/existing/existing is not valid. The parent node must pre- exist.

Usage

Update operation can used to perform the following actions:

- Re-parent/move category value(s) and sub-categories
- Retire category value(s)
- Change the Name(s) of category value(s)
- Add new category values
- Un-retire category values

Update *cannot* be used to update (or populate) existing Code fields (new category values with Codes will be inserted).

The XML hierarchy does not have to match the existing structure of category values, as code is used to identify the specified category values.

No action is taken on the category values that are not provided in the Update operation XML.

The rules of the Update operation are as follows:

1. Category values nodes that do not exist in the database (based on the code value) are considered adds, and will added to the hierarchy.
2. To add a new a new category value (using Update), the new node must be added as a child of a node of an existing category value. Following the Regions example used in the Insert and UpdateCodes syntax:
    a. Following the Regions example, the current structure of United States/Massachusetts/..can be modified to United States/NewEngland/Massachusetts/ Existing/new/existing is valid.
    b. The current structure can not be modified to North America/United States/Massachusetts/
3. Category values that do exist in the database, but are located in a different location in the XML hierarchy of the Update operation, are moved along with their sub-categories.
4. Category values provided in the XML and have corresponding records in the database that are retired will be un-retired. In all other ways (moving, etc.) they will be treated the same as the active category values.
5. Moving category values is allowed only within the same Category. It is not allowed to move a Location to a Job Type.
6. Retiring Category Values - a Category node may contain a Retire node, which in turn may contain one or more CategoryValue node(s). It is not necessary to provide a hierarchy within the Retire node, as every specified CategoryValue node will be retired (or deleted, if not in use) along with its sub-categories.

> **Note**: The retire operation is always performed at the end of the update for the given Category.

> **Note**: To use the Retire node, all category values in the hierarchy must have a code value.

Example syntax

The following syntax introduces a new child node called "New England" and moves Massachusetts and its child nodes to a new parent region called "New England":

```
<?xml version="1.0" encoding="windows-1252"?>
<root>
    <Email/>
    <Version>1</Version>
    <Operation>Update</Operation>
    <Categories>
        <Category>
            <Name>Region</Name>
            <CategoryValue>
                <Name>All Regions</Name>
                <Code>ALL_REG</Code>
                <CategoryValue>
                    <Name>United States</Name>
                    <Code>US</Code>
                    <CategoryValue>
                        <Name>New England</Name>
                        <Code>US_NE</Code>
                        <CategoryValue>
                            <Name>Massachusetts</Name>
                            <Code>US_MA</Code>
                            <CategoryValue>
                                <Name>Bedford</Name>
                                <Code>US_MA_BED</Code>
                            </CategoryValue>
                            <CategoryValue>
                                <Name>Framingham</Name>
                                <Code>US_MA_FRA</Code>
                            </CategoryValue>
                        </CategoryValue>
                    </CategoryValue>
                </CategoryValue>
            </CategoryValue>
        </Category>
    </Categories>
</root>
```

The following syntax retires the Region category:

```
<?xml version="1.0" encoding="windows-1252"?>
<root>
    <Email/>
    <Version>2</Version>
    <Operation>Update</Operation>
    <Categories>
```

```
        <Category>
            <Name>Region</Name>
            <Retire>
                <CategoryValue>
                    <Name>All Regions</Name>
                    <Code>ALL_REG</Code>
                </CategoryValue>
            </Retire>
        </Category>
    </Categories>
</root>
```

Replace

Usage

The Replace operation requires ALL Code values to be populated in the existing hierarchy (the database).

Replace will retire (or delete, if possible), the current subtree of the specified top <CategoryValue> node, and create a new hierarchy based on the provided XML structure. In order to replace the entire hierarchy, every Category with corresponding top nodes (e.g. Location/All Locations) must to be specified.

Aside from initially retiring the specified subtree, Replace works identically to Update - retired category values that are specified in the XML will be un-retired and moved, and new ones will be created.

Example Syntax

The following example replaces Massachussetts (and it's child nodes) Conecticut with two new child nodes.

```
<?xml version="1.0" encoding="windows-1252"?>
<root>
    <Email/>
    <Version>1</Version>
    <Operation>Replace</Operation>
    <Categories>
        <Category>
            <Name>Region</Name>
            <CategoryValue>
                <Name>All Regions</Name>
                <Code>ALL_REG</Code>
                <CategoryValue>
                    <Name>United States</Name>
                    <Code>US</Code>
                    <CategoryValue>
                        <Name>New England</Name>
                        <Code>US_NE</Code>
                        <CategoryValue>
                            <Name>Connecticut</Name>
                            <Code>US_CT</Code>
                            <CategoryValue>
```

```
                        <Name>Hartford</Name>
                        <Code>US_CT_HAR</Code>
                    </CategoryValue>
                    <CategoryValue>
                        <Name>New Haven</Name>
                        <Code>US_CT_NWH</Code>
                    </CategoryValue>
                </CategoryValue>
            </CategoryValue>
            <CategoryValue>
                <Name>North Carolina</Name>
                <Code>US_NC</Code>
                <CategoryValue>
                    <Name>Winston-Salem</Name>
                    <Code>US_NC_WS</Code>
                </CategoryValue>
                <CategoryValue>
                    <Name>Durham</Name>
                    <Code>US_NC_DUR</Code>
                </CategoryValue>
            </CategoryValue>
        </CategoryValue>
    </CategoryValue>
</Category>
</Categories>
</root>
```

RepaceAll

Usage

The ReplaceAll operation replaces the entire current hierarchy with the one specified in the XML.

The process deletes or retires all existing category values and the new structure is processed the same way as an Update, without the restriction that all Codes must be populated.

Categories specified in the XML that do not match any Categories in the database are created as part of the transaction.

ReplaceAll does not require all category values have Codes; therefore, the following matching logic is applied for category values specified in the XML:
If a Code is specified, the operation attempts to find a matching Code:

1. If found, the category value is moved, updated, and un-retired (if needed)
2. If not found, next try to match category value by name and parent (as specified by the XML structure)
    a. If found, the category value is un-retired (if needed) and updated (including Code)
    b. If not found, the category value is treated as a new category value and is added.

Permissions

"ManageCategoryUpload" permission is required to access the Import/Export Category Values page and perform Import/Update/Export Category Values operations.

ManageCategoryUpload permission implicitly grants the ManageCategory annd ManageCategoryValue permissions.

ManageCategoryValues permission is required for any changes to Category Values.

Output

If the Submit of an upload document is successful, no message is displayed on the Import/Export Category Values page.

- You may immediately review the results of the upload by choosing Manage Categories to view the Category hierarchy.
- If you choose to regress your change, modify your XML upload file, and try again, you can easily do so by choosing to Revert the corresponding Restore Point version.

The email notification is configured through the Administration→ Notification option in the Onboarding interface. The message Types (under "Bulk Upload") are:

- Bulk Category Value Upload Results
- Erroneous Bulk Category Value Upload

The default notification references two available email addresses to for the notification configuration:

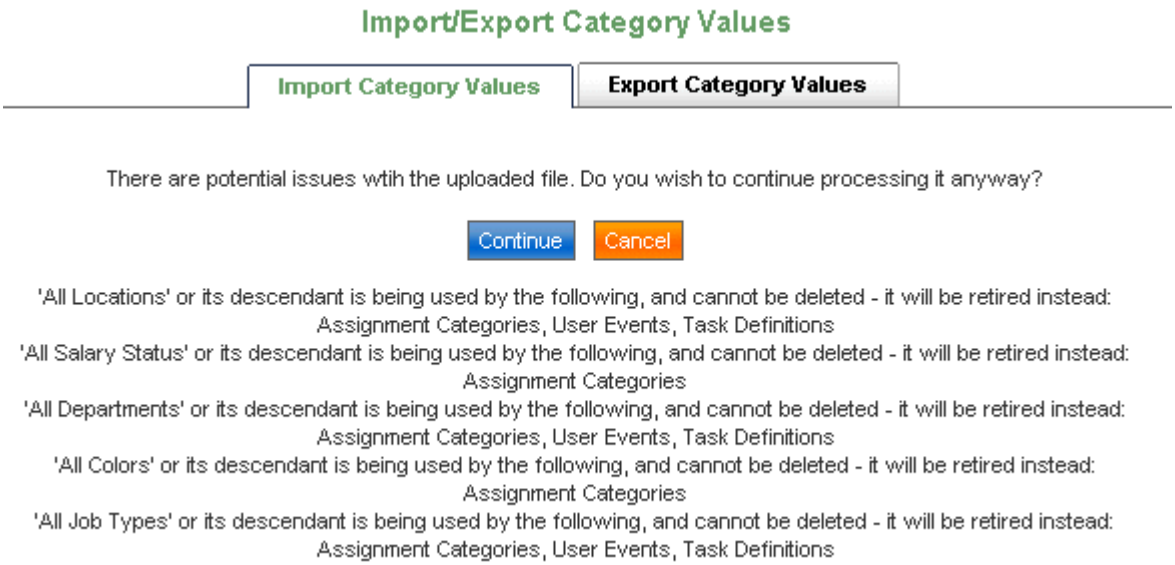
1. [$BULK_UPLOAD_MANAGER_EMAIL$] populated by the Bulk Upload Manager Email Address configuration setting
2. [$SPECIFIED_EMAIL_RECIPIENTS$] populated by the <Email> tag passed in the uploaded XML file.

An example default message is as follows:

```
The NewCategory.xml starting at 2:34:43 PM Thursday, June 25, 2009 and ending at: 2:34:43 PM
Thursday, June 25, 2009 has the following results:

Number of category values Created: 7
Number of category values Failed: 0


Follow this link to log in to Onboarding: http://server3:85/rc/Default.aspx

Thank You
```

Retire confirmation

If the hierarchy is modified to remove categories being referenced in active events, tasks, and employee assignments, Submit generates a confirmation message. The example below used ReplaceAll to minimally replace the existing hierarchy.



Choose Continue to execute the uploaded file or Cancel to cancel the Submit.

Error messages

If an error exists, one of the following text message is displayed upon Submit:

- Failed to create restore point.
- 'Code' node doesn't exist for <name of Category Value>
- Input structure doesn't match database structure for <Name of Category>
- Code is too long for {0}
- Malformed category upload XML
- Category Value with code '<Code>' doesn't exist.
- Code must be specified to retire Category Value '<Category Value name>'
- Invalid Portal Role: '<Eprise role>'
- 'Update' and 'Replace' require all Category Value Codes to be populated and are not supported in your environment. Please use 'ReplaceAll' to bulk update your category values.
- '<Category Name>': only one root Category Value is allowed per Category
- Internal Code '<Code>' already exists. Internal Codes must be unique if specified.
- A Category with the name '<Category Name>' already exists.
- Category '<Category Name>' does not exist.

Working with categories

Overview

Onboarding category upload allows Onboarding administrators to import and update category values and modify the hierarchy of the category structure. In addition to making integrations easier, the "Import and Export Category Management" feature eases maintenance of category value.

Prior to Onboarding v 2.3 administrators could create and updated categories and category values through the **Manage Categories** menu option from the user interface. Also through the interface an administrator could import category values using XML syntax through the **Import Category Values** menu option.

Onboarding v 2.3 significantly expanded the features of the XML import syntax. Administrators can now update category values, and replace category values. The replace functions allow administrators to move child nodes and restructure the category value tree.
The expansion of the category value update features also introduced two supporting features to support accurate category hierarchy modifications:

- Export of the current structure and
- Restore point functionality, allowing an administrator to both preview and regress a category hierarchy change

In addition to applying the XML file through the Import Category Values page in the Onboarding user interface, a new web method has been added to support these category value changes.

Available upload features

The table below provides a high level overview of each of the update operations. The specific syntax and example for each operation are provided later.

Each function is specified by an operation node to implement a change. Some of the operations have prerequisites in the data to be applied. The prerequisites are listed in the Requirements column.

Operation	Description	Requirement for the operation to be applied
Import	Add new category values	Category must exist (Category Name)
Update	Re-parent (move) category value(s) and sub-categories, Retire category value(s), Un-retire category values, Change the Name(s) of category value(s), and Add new category values	**All** category code values must exist
UpdateCodes	Update codes of existing categories	An exact match must exist between the UpdateCodes XML and the existing category hierarchical structure
Replace	Retire (or delete, if possible), the current subtree of the specified top CategoryValue node, and create a new hierarchy based on the provided XML structure.	An exact match must exist between the Replace XML and the existing category hierarchical structure. Every Category with corresponding top nodes (e.g. Location/All Locations) must to be specified
Replace All	Replace the entire current hierarchy with the specified structure in the XML. Categories specified in the XML that do not match any categories in the existing hierarchy will be created as part of the transaction.	None

Hierarchical structure of category values

Onboarding category values are available in a hierarchical tree view structure. The parent node is the category name with one or more child nodes communicating the category values.

An example category tree structure for a category called "Locations" is has the following category values:

All Locations

- United States
  - Massachusetts
  - North Carolina
  - Florida
- Canada
  - Edmonton
  - Toronto
- Europe
  - Germany
  - England

The hierarchy of category values is a traditionally parent / child tree structure with the parent node including child nodes.

In the example, if an employee is assigned to the United States, then employee is eligible for tasks for Massachusetts, North Carolina, and Florida.

The XML of the upload record mimics the graphical hierarchy in the parent/child relationship.

Hierarchical Structure of Categories

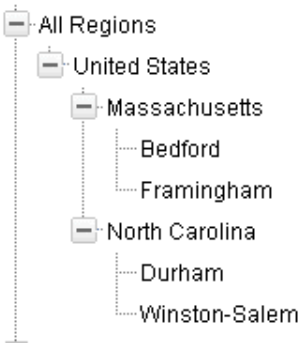
Category syntax

The hierarchical category tree has an indefinite number of child categories, which allows you to create a descriptive hierarchy to reflect your organization or mirror an existing system. The granularity of your category definitions is not restricted by Onboarding.
The structure mimics the entry of category values through the Onboarding interface.

- The high level parent node is defined by category name. The category name must be created through the Onboarding user interface.
- Each category value is minimally defined by a Name node, and optionally defined by a Name, Code, and Eprise role. Code and Eprise Role may be blank. "Eprise Role" is synonymous to "Portal Role".
- Child nodes are created by nesting category values.

Example structure

In all cases the high level parent node must be created through the Onboarding user interface (Manage Categories option).
In the following example with nested values, the high level parent node called "Region" was created through the Manage Categories
-> Add Category option.

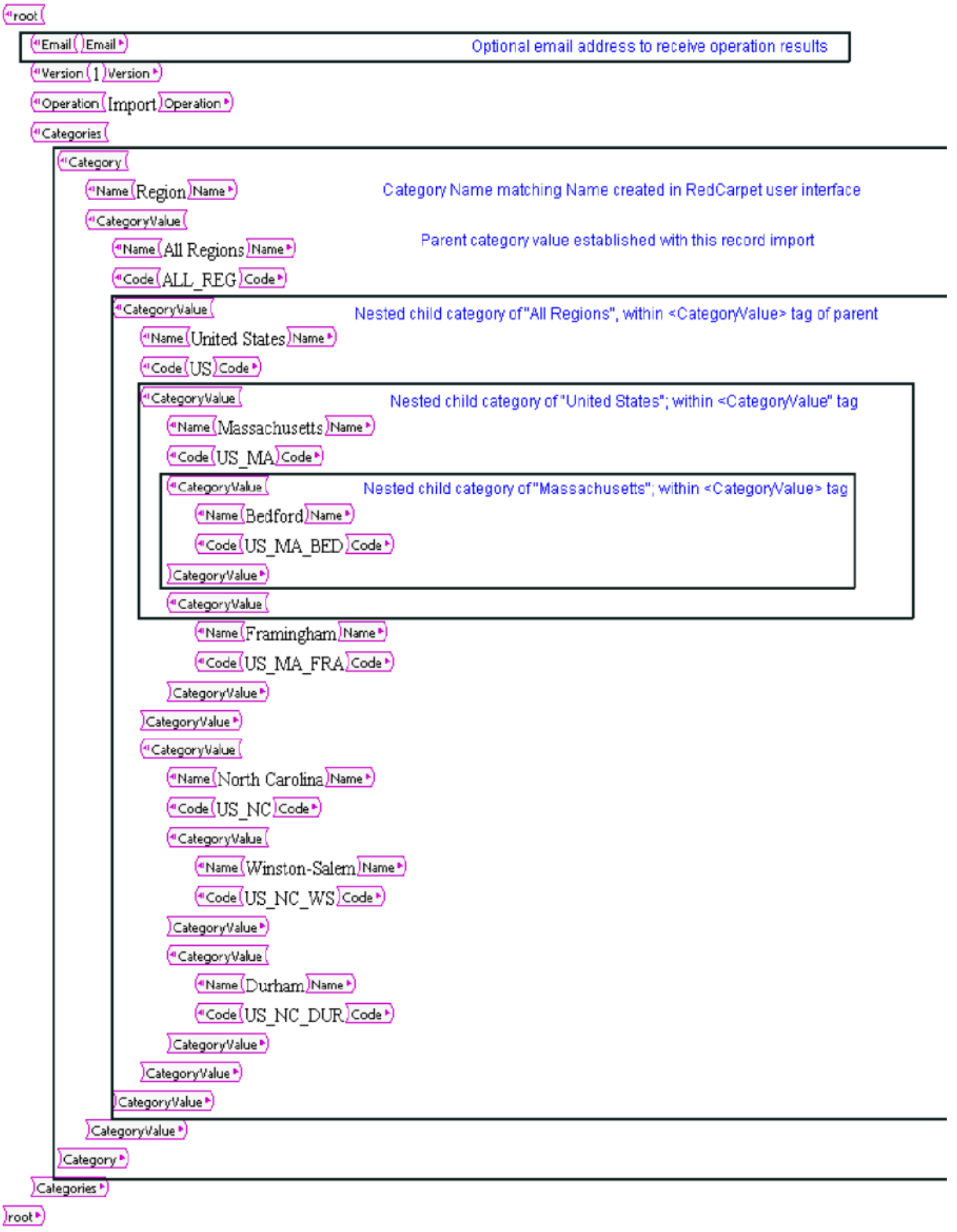


The following XML creates the structure:

```
<?xml version="1.0" encoding="windows-1252"?>
<root>
    <Email/>
    <Version>1</Version>
    <Operation>Import</Operation>
    <Categories>
        <Category>
            <Name>Region</Name>
            <CategoryValue>
                <Name>All Regions</Name>
                <Code>ALL_REG</Code>
                <CategoryValue>
                    <Name>United States</Name>
                    <Code>US</Code>
                    <CategoryValue>
                        <Name>Massachusetts</Name>
                        <Code>US_MA</Code>
                        <CategoryValue>
                            <Name>Bedford</Name>
                            <Code>US_MA_BED</Code>
                        </CategoryValue>
                        <CategoryValue>
                            <Name>Framingham</Name>
                            <Code>US_MA_FRA</Code>
                        </CategoryValue>
                    </CategoryValue>
                    <CategoryValue>
                        <Name>North Carolina</Name>
                        <Code>US_NC</Code>
                        <CategoryValue>
                            <Name>Winston-Salem</Name>
                            <Code>US_NC_WS</Code>
                        </CategoryValue>
                        <CategoryValue>
                            <Name>Durham</Name>
                            <Code>US_NC_DUR</Code>
                        </CategoryValue>
                    </CategoryValue>
```

```
                </CategoryValue>
            </CategoryValue>
        </Category>
    </Categories>
</root>
```

The following graphical diagram describes the XML structure:


root
Email Email
Optional email address to receive operation results
Version 1 Version
Operation Import Operation
Categories
Category
Name Region Name
Category Name matching Name created in RedCarpet user interface
CategoryValue
Name All Regions Name
Code ALL_REG Code
Parent category value established with this record import
CategoryValue
Nested child category of "All Regions", within <CategoryValue> tag of parent
Name United States Name
Code US Code
CategoryValue
Nested child category of "United States"; within <CategoryValue" tag
Name Massachusetts Name
Code US_MA Code
CategoryValue
Nested child category of "Massachusetts"; within <CategoryValue> tag
Name Bedford Name
Code US_MA_BED Code
CategoryValue
CategoryValue
Name Framingham Name
Code US_MA_FRA Code
CategoryValue
CategoryValue
CategoryValue
Name North Carolina Name
Code US_NC Code
CategoryValue
Name Winston-Salem Name
Code US_NC_WS Code
CategoryValue
CategoryValue
Name Durham Name
Code US_NC_DUR Code
CategoryValue
CategoryValue
CategoryValue
CategoryValue
Category
Categories
root

Category Value record field descriptions

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	✓	XML Document header	
<root>	✓	Parent element.	Occurs once per XML document.
<Email></Email>		Email address of a Onboarding administrator to receive success and failure messages.	This value populates the [$SPECIFIED_EMAIL_RECIPIENTS$] Occurs once per XML document
<Version></Version>		Version is populated on an export record. The version is not applied on an import.	Occurs once per XML document

Node	Required	Description	Notes
<Operation></Operation>	✓	Valid values are: • Import • UpdateCodes • Update • Replace • ReplaceAll See *Table: Overview of Operations* for each description.	Occurs once per XML document. Each operation is an atomic transaction. If errors occur during the operation, Onboarding will cancel the operation and revert back to the latest checkpoint (created prior to the start of the transaction). This applies to Update and Replace operations as they effect the hierarchy. UpdateCode does not cause a checkpoint to be created. No changes to the hierarchy or category values will be permitted while a Bulk Update operation is in progress
Categories	✓	Parent Node for all categories for current operation	Occurs once per XML document.

Node	Required	Description	Notes
Category	✓	Parent Node of Category Name and Category Values.	Occurs once for each Category affected by the operation
Name	✓	Child node of <Category> containing category name.	Category name is visible in the user interface on the Manage Localizations page. Occurs once per <Category> node.

Node	Required	Description	Notes
CategoryValue	✓	Occurs for each category value in the <Category> node.	• Child node of <Category> • Child node of <CategoryValue> to create a level of hierarchy in the tree structure. Each level of nesting created the hierarchy.
Retire		Parent node to retire all CategoryValue nodes it contains.	Applies only to Update operation

Node	Required	Description	Notes
Name	Required based on Operation type: • Import: ✅ • Update Codes: ✅ • Update • Replace • Replace All: ✅	Display name of the category value used throughout the Onboarding user interface.	The category value is visible in the category tree in the Onboarding interface text and CDATA are both valid

Node	Required	Description	Notes
Code	Required based on Operation type: • Import: • Update Codes: ✓ • Update: ✓ • Replace: ✓ • Replace All:	Unique category value code.	Code is not visible in the Onboarding user interface beyond the category management page. Code is useful to synchronize category data across multiple applications and to uniquely identify each category value for maintenance purposes. Update and Replace operations require **all** values in the tree have codes.
EpriseRole		String value matching a predefined portal role.	

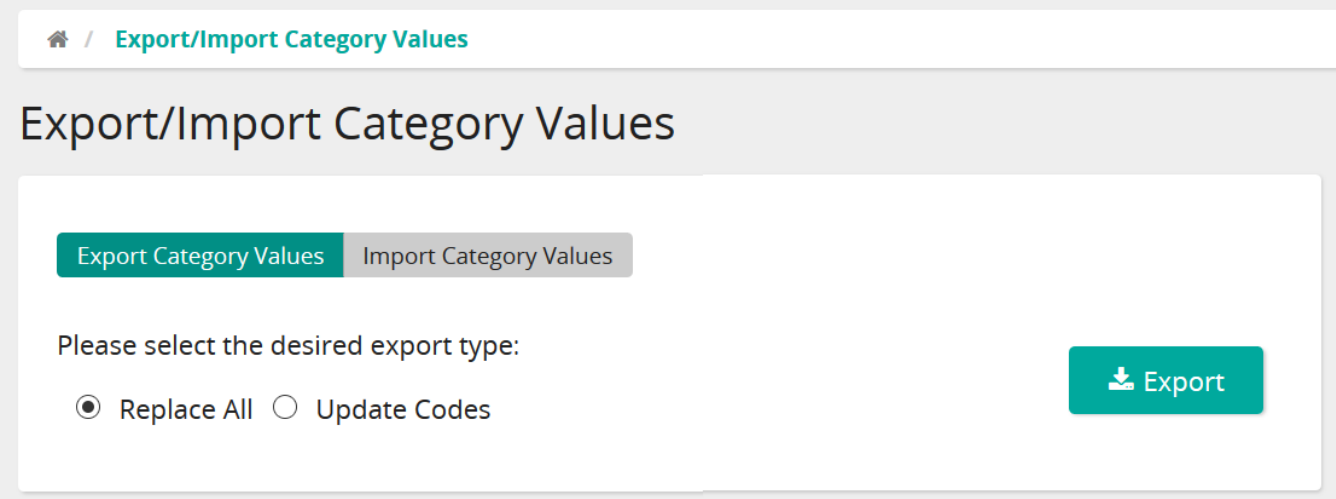
Export and Restore Points

Export

The second tab on the Import/Export Category Values page is the Export Category Values option. Both the UpdateCodes and the ReplaceAll operations are very dependent on the existing category value hierarchy.

- UpdateCodes requires an exact match on the hierarchy to populate the code as a unique identifier.
- ReplaceAll will *replace the entire hierarchy* with the hierarchy in the upload record

The export option is an easy starting point to use in these operations.



Both options export the entire category hierarchy.

- Choose Replace to generate an xml document containing the ReplaceAll operation.
- Choose Update Codes to generate an xml document containing the UpdateCodes operation.
- Click Export to create the file of XML containing the entire hierarchy.

Restore Points

A snapshot of the category hierarchy is available through the Manage Category page.The snapshot or "Restore Point" is available for an administrator to revert (regress) to a previous version of the category hierarchy.

A Restore Point is a useful support feature if the hierarchy is erroneously modified with XML that is correctly formed but incorrectly implemented.
RedCarpet will automatically create a restore point at the start of an upload. No restore point is created for Update Codes operations, as they do not affect the structure of the hierarchy.

A restore point may also be manually created by an administrator.
Because the restore point creation contains an Export operation, the ManageCategoryUpload privilege is required.

Using Restore Points

1. In SilkRoad Onboarding, choose the **Show Restore Points** option on the Manage Categories Page.
2. Choose **Revert** to restore an existing restore point. The restore will be a ReplaceAll operation, and will obey the rules specified above.
3. Choose **Preview** to preview the corresponding XML for the hierarchy version of the restore point. A preview is an export of the hierarchy.
4. Choose **Delete** to remove the restore point from the database completely.

FieldValueUpload

Usage

Allows a developer to insert, update, or replace descriptions and values in eForms lists.

The functionality of this method is available through the Onboarding user interface under the eForms Administration.

Parameters

Parameter	Required	Description	Type
strSecurityToken	☐	Valid Session ID for consuming service	simple string
strXML	☐	XML is described below.	simple string

FieldValueUpload strXML nodes

Node	Required	Description	Notes
<?xml version="1.0" encoding="utf-8"?>	✓	XML Document header	
<Import>	✓	Parent node occurs once	
<Operation> </Operation>		Edit Operation occurs once	Possible values: AddOrUpdate (default) • Adds and updates items in the named list Replace • Replaces all items in the named list Example Syntax: <Operation>Replace</Operation>

Node	Required	Description	Notes
<List>	✓	Named list can occur multiple times	
<Name></Name>	✓		Text list name. If name does not exist the list will be created. If the name exists the list will be updated or replaced based on the operation. Example Syntax: <Name>Size</Name>
<Item> <Value></Value> <Text></Text> </Item>		An item occurs for each selection value / description pair available in the selection list.	Example syntax: <Item> <Value>S</Value> <Text>Small</Text> </Item>
</List>	✓	End List for each List	
</Import>	✓	End Import	

Returns

An XML string of type EpsStringEx.

An XML document defined as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EpsStringEx xmlns="http://Eprise">
 <ErrorString>No Error</ErrorString>
```

```
  <ErrorNum>1</ErrorNum>
  <Data>
   <![CDATA[ XML document with results]]>
  </Data>
 </EpsStringEx>
```

<ErrorString>

Return code description. The value of *ErrorNum* should be checked for a the value not equal to 1 to determine an error description vs. a success message.
"" or "*xxxx*" — return code description

<ErrorNum>

This variable indicates success or failure. Values are:

1 (one) — method succeeded

0 (zero) — method failed

<Data>

Examples

Example of creating a list

Example to create a list called "Size" with 3 rows: (Add or Update Assumed)

```
<?xml version="1.0" encoding="UTF-8"?>
<Import>
 <List>
  <Name>Size</Name>
  <Item>
   <Value>S</Value>
   <Text>Small</Text>
  </Item>
  <Item>
   <Value>M</Value>
   <Text>Medium</Text>
  </Item>
  <Item>
   <Value>L</Value>
   <Text>Large</Text>
  </Item>
 </List>
</Import>
```

Example of adding a value

Example to add a row to an existing list called "Size": (Add or Update Assumed)

```
<?xml version="1.0" encoding="UTF-8"?>
<Import>
```

```
 <List>
 <Name>Size</Name>
  <Item>
   <Value>XL</Value>
   <Text>Extra Large</Text>
  </Item>
 </List>
</Import>
```

Example of replacing a value

This example replaces the values on an existing list called "Size": (Replace specified)

```
<?xml version="1.0" encoding="UTF-8"?>
<Import>
 <Operation>Replace</Operation>
 <List>
  <Name>Size</Name>
  <Item>
   <Value>1</Value>
   <Text>Small</Text>
  </Item>
  <Item>
   <Value>2</Value>
   <Text>Medium</Text>
  </Item>
  <Item>
   <Value>3</Value>
   <Text>Large</Text>
  </Item>
 </List>
</Import>
```

Using Transformations

Onboarding methods allow you to use XSLTs to transform both the input and the output of many Onboarding methods.

Transforming the XML from one format to another is useful for changing export from one system to the expected the input format of a Onboarding method. Onboarding allows a transformation of the output to another format for consumption of the output to another system.

SilkRoad also uses the output XLSTs to maintain backward compatibility with previous versions of the Onboarding methods.

Manage transformations

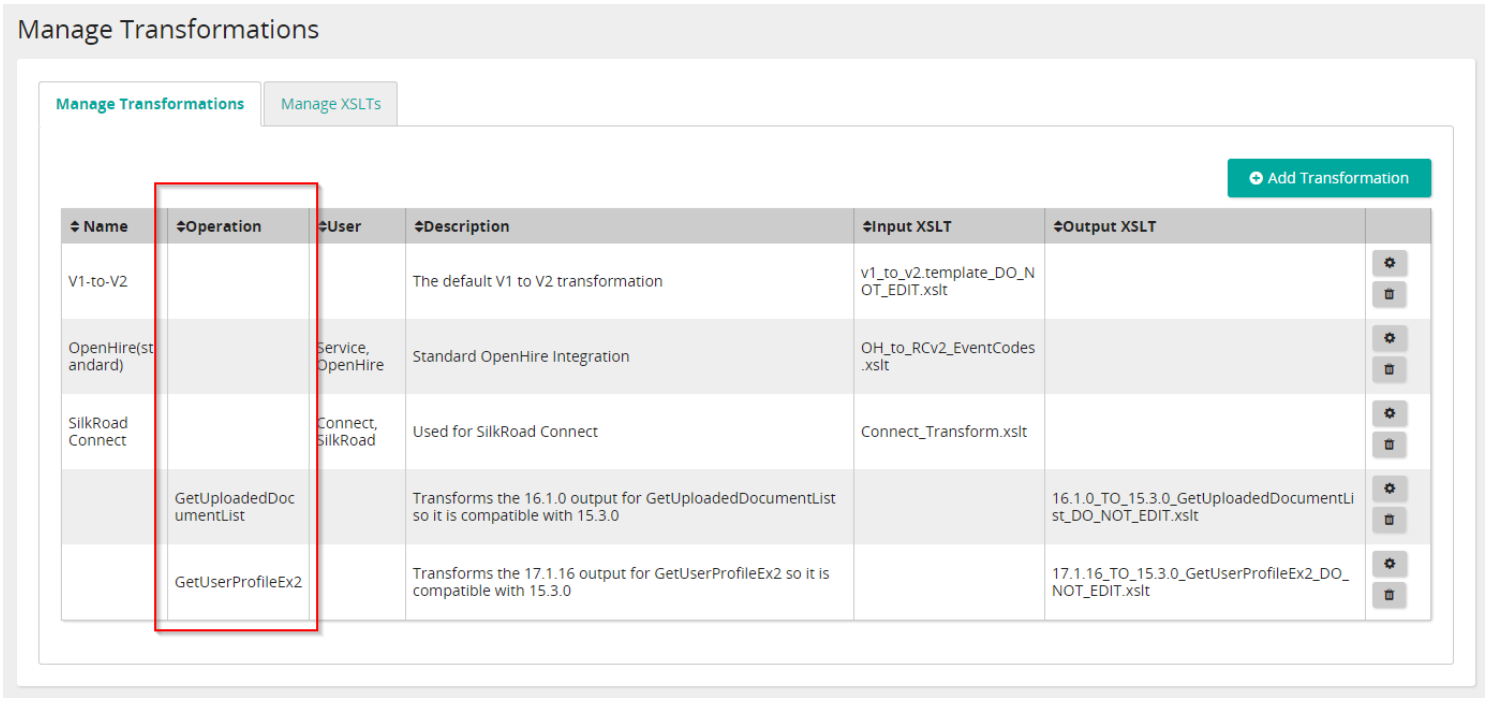
Usage

The Manage Transformations page is available through the Onboarding user interface. It allows you to upload XSLT files into the Onboarding database. Uploaded files can be referenced by the method call either by operation or by reference. Onboarding administrators with the "Manage Transformation" privilege have access the Manage Transformations Administration menu option. Defined transformations are applied based on reference by name each time an operation (method) is executed or which user is executing the method.

Default configuration

Version 2.8.0 introduced the first set of updates to existing method outputs. The updates are in response to customer requests to include more information on the successful records for bulk operations. By default, the output of the method will match the 2.7.x output. The default behavior does not change to insure system integrations with Onboarding will work after upgrading to versions after 2.7.x.

To take advantage of updates to the output, you must modify the transformation definition to remove the transformation of 2.8.0 output to 2.7.x output.



If an XSLT is managed by Operation, each time the operation is executed the XSLT will be applied. For example, after upgrade to 2.8.0 each time BulkUserUpload, LaunchEvent, or Update Event is executed the default transformation is applied to remove 2.8.0 nodes from the output.

To take advantage of 2.8.0 output, you should modify the XLSTs from the default operation. The recommended technique to do this is to edit the manage transformation criteria to reference the XSLT by name instead of operation.

- defining the XSLT by operation applies the XSLT each time the method is executed
- defining the XLST by name applies the XSLT only when the XML (passed as an input parameter) includes the node

```
<TransformationName></TransformationName>
```

Methods supporting transformations

The following Onboarding methods support input and output transformations. Input transformations expect the resulting transformation to result in the an XML document that matches the method XSD.

- BulkUserUpload
- XMLUserEdit
- LaunchEvent
- UpdateEvent
- CategoryUpload
- GetUserIDList
- GetUserIDList
- GetTasks
- GetUserProfileEx2
- AddTasksEx2
- GetCompletedFormsList
- GetUploadedDocumentList
- CompleteTaskEx
- ReopenTask
- AddTaskNoteEx
- AssignTask
- GetFormXMLEx
- FieldValueUpload

Applying transformations rules

1. Each transformation allows for new optional property called Operation. This may be set to the name of one of a list of supported web methods, if needed. In 2.8.0, the list of supported methods are available. The supported methods are BulkUserUpload, XmluserEdit, LaunchEvent, UpdateEvent, and CategoryUpload.
    - GetFormXmlEx allows for an additional optional property called Form Type. Form type allows you to specify a transformation per form.
2. When handling all new API operations, Onboarding determines whether or not to use a transformation and which one to use based on the following rules:
    - If a Transformation name specified in a node in the input XML,
        - Onboarding searches for the transformations matching that name, and any of Operation (the web method being handled) or Logged in user
        - Onboarding uses the transformation that matches most criteria (Operation AND Logged in user, followed by only logged in user, followed by only Operation).
    - If no Transformation name specified in a node in the input XML,

    - Onboarding searches for the transformations matching any of Operation (the web method being handled) or logged in user.
    - Onboarding uses the transformation that matches most criteria (Operation AND Logged in user, followed by only logged in user, followed by only Operation).
- When GetFormXmlEx is the web method being handled, whenever Onboarding matches Operation above, Onboarding first matches Operation and Form Type and then only Operation.

Appendix: Error Codes

BulkUserUpload error codes

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The SSOAuthParam value is already in use by another user	21022	Attempt to add a user when specified AuthParam or SSOAuthParam is used by an existing user
Invalid Input	The Employee_HRISID is already in use by another user	20125	Attempt to add a user when specified Employee_HRISID is used by an existing user
Invalid Input	The LifeSuiteID value is already in use by another user. Attempt to add a user when specified LifeSuiteID is used by an existing user	20126	RESERVED For SilkRoad
Event Already Exists	X event already exists for user:Y	30000	One "In Progress" event (per event type) at a time.
Fatal Error	Fatal Error	30001	Caused by malformed XML
Fatal Error	Failed to add event to user because the sessionid executing the method is executing the method is	30002	Not launched by the event coordinator
Fatal Error	Unknown event name	30003	
Invalid Input	Invalid category value name for *named category:value*	30004	The value in passed for <Type>Name</ Type> did not match a valid category name.
Invalid Input	Invalid category value name for *named category:value*	30005	The value in passed for <Type>Code</ Type> did not match a valid category Code.
Invalid Input	Invalid category value name for *named category:value*	30006	The value in passed for <Type>Path</ Type> did not match a valid category Path.

<Type>	<Message>	<ErrorCode>	Description
Event Error	Named category does not exist on the Event definition	30007	Event values launched for imported user must map to field defined on the event definition
Event Error	Named person does not exist on the Event definition	30008	
Invalid Input	Missing abstract category node under category node	30015	The <Category> node is missing the <Name> child node.
Invalid Input	Missing category value node under category node	30016	The <Category> node is missing the <Value> child node
Invalid Input	Missing category value node under category node	30017	The <Category> node is missing the <Type> child node
Invalid Input	Unknown person (manager) name	30018	
Invalid Input	Named person does not exist	30019	
Invalid Input	Failed to update event because the sessionid executing the method is not an event coordinator	30020	
Invalid Input	Unknown date name	30021	
Invalid Input	Named date does not exist on the event definition	30022	
Invalid Input	Date format is invalid	30023	Expected date format is YYYY-MM-DD
Invalid Input	More than one category value name found for value category: *category name*	30024	<Type>Name</Type> was specified but Name is not a unique category identifier
Invalid Input	No user identification node could be found for the {0} event node	30025	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Missing person (manager) name	30026	
Invalid Input	Missing person (manager) value	30027	Applies to all missing <Person> nodes. The specific <Name/> node is not specified. .
Invalid Input	Missing date name	30028	
Invalid Input	Missing date value	30029	
Invalid Input	Missing category name node under category node	30030	Each <Category> occurence must have a <Name> node
Invalid Input	Invalid category value type (Type entered: *x*) must be 'Name', 'Code', or 'Path'	30031	Valid <Type> values are: • <Type>Name</Type> • <Type>Code</Type> • <Type>Path</Type>
Invalid Input	Invalid event name '*x*' found	30032	Undefined <Event><Name> value was specified.
Invalid Input	Parsing failed due to: *x*	30033	Unrecognized node specified.
Invalid Input	Unknown date name: *x*	30034	Undefined <Date> value specified
Invalid Input	Invalid manager identifier entered for *x* manager: {1}	30035	Undefined manager name specified on event or unidentified ApprovalManager specified on a pending user
Invalid Input	Invalid manager name entered: {0}");	30036	Invalid <Value> for the coorsponding <Person> <Name>
Invalid Input	Invalid date value entered for *x* date, must be in YYYY-MM-DD format: {1}	30037	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	User marked as ForceActiveNewHire, but does not have required *x data*	30038	Required fields must be included in the ForceActiveNewHire record.
Invalid Input	Missing category name node under category node	30039	
Invalid Input	The EmployeeIDType {0} entered for {1} is unknown. Please use one of the following: LifeSuiteID, Employee_HRISID, Email, LoginID, GUID, or SSOAuthParam.	30057	
Fatal Error	No users nodes found in file.	31000	No data.
Invalid Input	Termination Date not in proper YYYY-MM- DD format: *date*	31001	Invalid date format
Event Error	Users can be uploaded with 1 event at most. The *named* event must be removed.	31002	Bulk Import is restricted to one event per new user. Any additional events must be removed from the bulk import record. Additional events can be executed using the LaunchEvent method.
Event Error	Event node found but missing name	31003	<Event> tag specified without a <Name> tag
Event Error	Invalid event name *name* found	30003	Undefined Event name
Invalid Input	Unexpected *X* node found under *Y* node	31004	Malformed XML syntax where *X* and *Y are specified nodes.*
Invalid Input	Invalid role value: *value*	31006	Invalid team value specified
Invalid Input	Cannot create user from parsed data, despite the forceactivenewhire flag, because the user's data had errors	31007	Data error in user profile information.

<Type>	<Message>	<ErrorCode>	Description
User Validation	Ignoring unknown value: *key=value*	31008	Invalid node syntax
User Validation	Failed to create active user due to errors in the user information	31009	Data error in user profile information.
Event Error	Failed to create active user due to errors in the user event information	31010	
Invalid Input	Missing eVerify status data	31011	<eVerifyCaseNumber> and <eVerifyStatus> must both have values if either have values.
Invalid Input	Missing eVerify case number	31012	<eVerifyCaseNumber> and <eVerifyStatus> must both have values if either have values.
User Validation	message (exception)	31013	
User Validation	property : message	31014	
User Save Error	Failed to save a user due to a database error (message)	31015	Malformed XML caused database error.
User Save Error	Application exception while saving: message (Inner exception: inner message)	31016	Call SilkRoad support.
User Save Error	Exception while saving: *exc*	31017	
User Save Error	Failed to create pending user	31018	

<Type>	<Message>	<ErrorCode>	Description
User Save Error	The user invoking the bulk import does not have the privilege to create pending users.	31019	Incorrect privileges of the user executing the method
Invalid Input	The user invoking the bulk import does not have the privilege to create active users.	31035	Employee who obtained the SessionID must be on a team with the "Create User" privilege.
	XML user edit is stopping because an error was found and XMLUploadStopOnError is set to 1	31036	XMLUploadStopOnError set through the Administration ->Settings page. Set XMLUploadStopOnError to False to allow user edit to continue modifying valid records.
Invalid Input	The e-mail address (*x*) which was specified for the *n* user node is not used by any user in the system	31037	Invalid email address specified.
Invalid Input	The e-mail address (*x*) which was specified for the {*n*} user node is used by {*nn*} users so it cannot be used to uniquely identify a single user	31038	Non unique email address specified.
Invalid Input	An unkown error has occured	31039	
Invalid Input	Failed to load XML document (*x*)	31040	
Invalid Input	XML user edit failed due to:{*x*}	31041	
Invalid Input	No user could be found for the {*x*} user node	31042	
Invalid Input	Unkown abstract category: {*x*}	31043	Unknow "Alias Name" category reference specified.
Invalid Input	The EmployeeIDType {0} entered for {1} is unknown. Please use one of the following: LifeSuiteID, Employee_HRISID, Email, LoginID, GUID, or SSOAuthParam.	31053	EmployeeIdType and ID are both provided, and type is unknown, for a named person on a key property

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Insufficient Privileges to perform the action	34001	User executing script does not have the correct privilege (team membership or event coordinator designation) to execute the referenced action
Invalid Input	{0} cannot be used because it is not a leaf category value (i.e. It has child category values)	36015	
Invalid Input	Key Property: Invalid Category Name - 'LOCATION'	39001	
Invalid Input	Key Property: Invalid Category Code - 'CATEGORY_1'	39002	
Invalid Input	Key Property: Invalid Person Name - 'MANAGER'	39003	
Invalid Input	Key Property: Invalid Person Code - 'PERSON_1'	39004	
Invalid Input	Key Property: Invalid Date Name - 'START'	39005	
Invalid Input	Key Property: Invalid Date Code - 'DATE_1'	39006	
Invalid Input	The following key properties are required but do not have a value: 'LIST OF MISSING REQUIRED KEY PROPERTIES'	39007	
Invalid Input	Key Property: Invalid category value name for 'LOCATION' category: 'FAKE TOWN'	39008	
Invalid Input	Key Property: Invalid category value path for ''LOCATION category: ''LOCATION category: 'FAKE/PATH/4U'	39009	
Invalid Input	Key Property: Invalid category value code for 'LOCATION' category: 'FAKE_CODE'	39010	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Key Property: Invalid manager identifier entered for Manager manager:	39011	
Invalid Input	Key Property: Date format is invalid	39012	Expected format is YYYY-MM-DD
Invalid Input	Missing ID for a named person	39014	
Invalid Input	Missing EmployeeIDType for a named person	39015	

XMLUserEdit Error Messages

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The SSOAuthParam value is already in use by another user	21022	Attempt to modify the AuthParam or SSOAuthParam to a value used by an existing user NOTE: this errorcode value was 0 prior to 2.8.0. You must disassociate the default XMLUserEdit transformation to return this errorcode
Invalid Input	The Employee_HRISID is already in use by another user	20125	Attempt to modify the Employee_HRISID to a value used by an existing user NOTE: this errorcode value was 0 prior to 2.8.0. You must disassociate the default XMLUserEdit transformation to return this errorcode

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The LifeSuiteID value is already in use by another user	20126	RESERVED For SilkRoad Attempt to modify the LifeSuiteID to a value used by an existing user NOTE: this errorcode value was 0 prior to 2.8.0. You must disassociate the default XMLUserEdit transformation to return this errorcode
Fatal Error	Parsing failed due to: *message*	30001	Caused by malformed XML
Invalid Input	The e-mail address used to identify a user is used by more than one user	30013	
Invalid Input	"{0} event doesn't exist for user: {1}"	30014	0 - Name of the event in the default lanauge 1 - Name of the user
Invalid Input	Missing category name node under category node	30030	Each <Category> occurence must have a <Name> node
Invalid Input	Invalid category value type (Type entered: *x*) must be 'Name', 'Code', or 'Path'	30031	Valid <Type> values are: <Type>Name</Type> <Type>Code</Type> <Type>Path</Type>
Invalid Input	Invalid event name '*x*' found	30032	Undefined <Event><Name> value was specified.
Invalid Input	Parsing failed due to: *x*	30033	Unrecognized node specified.
Invalid Input	Unknown date name: *x*	30034	Undefined <Date> value specified
Invalid Input	Invalid manager identifier entered for *x* manager: {1}	30035	Undefined manager name specified on event
Invalid Input	Invalid manager name entered: {0}");	30036	Invalid <Value> for the coorsponding <Person> <Name>

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Invalid date value entered for *x* date, must be in YYYY-MM-DD format: {1}	30037	
Invalid Input	User marked as ForceActiveNewHire, but does not have required *x data*	30038	Required fields must be included in the ForceActiveNewHire record.
Invalid Input	Termination Date not in YYYY-MM-DD format: *date*	31001	Invalid date format
Invalid Input	The e-mail address (address) which was specified for the *nith* user node is not used by any user in the system	30012	User identified by email address can not be found
Invalid Input	The e-mail address (address) which was specified for the *nith* user node is used by n users so it cannot be used to uniquely identify a single user	30013	email address is not a unique identifier in your Onboarding installation.
Invalid Input	{0} event doesn't exist for user: {1}	30014	0 - Event Name 1 - Employee name
Invalid Input	More than one category value name found for value category: *category name*	30024	<Type>Name</Type> was specified but Name is not a unique category identifier
Invalid Input	Missing eVerify status data	31011	<eVerifyCaseNumber> and <eVerifyStatus> must both have values if either have values.
Invalid Input	Missing eVerify case number	31012	<eVerifyCaseNumber> and <eVerifyStatus> must both have values if either have values.
Fatal Error	XML user edit failed due to: *message*	31020	
Unknown User	No user could be found for the *ith* user node	31021	<User> node found with no child nodes
User Validation	No user identification node could be found for the *ith* user node	31022	Invalid UserToEdit_GUID was specified.

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	You cannot retire a retired employee	31023	
Invalid Input	You cannot unretire an employee who is not retired	31024	
Invalid Input	Employee could not be removed as a controller from the named team because that employee is not a controller of the team	31025	
Invalid Input	Employee could not be removed as a member from the named team because that employee is not a member of the team	31026	
Invalid Input	User could not be added to the named team because you are not a controller for that team	31027	
Invalid Input	User could not be removed from the named team because you are not a controller for that team	31028	
Invalid Input	The team named *name* is unknown	31029	
Invalid Input	Cannot add employee to assignment category because the employee is already assigned to that category	31030	
Invalid Input	Cannot remove employee from assignment category because the employee is not assigned to that category	31031	
Invalid Input	The named field is not editable or unknown. Because the XmlEditStopOnNonEditableField setting is enabled this user will be skipped.	31032	
Invalid Input	The named field is not editable or unknown.	31033	

<Type>	<Message>	<ErrorCode>	Description
User Save Error	Failed to save user: *message*	31034	
Invalid Input	The user invoking the bulk import does not have the privilege to create active users.	31035	Employee who obtained the SessionID must be on a team with the “Create User” privilege.
Invalid Input	XML user edit is stopping because an error was found and XMLUploadStopOnError is set to 1	31036	XMLUploadStopOnError set through the Administration ->Settings page. Set XMLUploadStopOnError to False to allow user edit to continue modifying valid records.
Invalid Input	The e-mail address (x) which was specified for the n user node is not used by any user in the system	31037	Invalid email address specified.
Invalid Input	The e-mail address (x) which was specified for the {n} user node is used by {nn} users so it cannot be used to uniquely identify a single user	31038	Non unique email address specified.
Invalid Input	An unkown error has occured	31039	
Invalid Input	Failed to load XML document (x)	31040	
Invalid Input	XML user edit failed due to:{x}	31041	
Invalid Input	No user could be found for the {*x*} user node	31042	For example: if an invalid LoginID, Employee_HRISID (no old syntax), SSOAuthParam (old syntax - AuthParam), Guid, LifeSuiteID the return will be: <Error> <Type>Invalid Input</Type> <Message>No user could be found for the 1st user node</Message> <Code>31042</Code> </Error>

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Unkown abstract category: {*x*}	31043	Unknow "Alias Name" category reference specified.
Invalid Input	The UserToEdit_*x* identifier node for the *y* user cannot be blank	31044	Employee identifier included in *y* (number) the record is blank.
Invalid Input	Can't retire user:	31051	The employee being retired is a named person (employers) referenced on active events or not completed tasks. You must reassign the employer's task and/or events and try to retire again.
Invalid Input	Purge Date must be at least 24 hours in the future.	31063	
Invalid Input	Purge Date not in YYYY-MM-DD format:	31062	
Invalid Input	Purge Reason must be a number. Valid values are: 0:TerminatedEmployee,1:ExternalSystem,2:GDPR,	31064	
Invalid Input	Missing required information: Terminate Event: You must specify an Event Name or Code as the first sub-node of the TerminateEvent node.	3400	
Invalid Input	Insufficient Privileges to perform the action	34001	User executing script does not have the correct privilege (team membership or event coordinator designation) to execute the referenced action
Invalid Input	The I-9 Administrator for bulk upload modifications setting is not set. Updating a user's Hire Date on an I- 9 through bulk upload requires this setting	35011	If an employee has an I-9 form a <HireDate> update affects the I-9 form section 2 begin employment date. A required configuration settting has not been set. See "Hire Date Updates" for additional information.
Invalid Input	Key Property: Invalid Category Name - 'LOCATION'	39001	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Key Property: Invalid Category Code - 'CATEGORY_1'	39002	
Invalid Input	Key Property: Invalid Person Name - 'MANAGER'	39003	
Invalid Input	Key Property: Invalid Person Code - 'PERSON_1'	39004	
Invalid Input	Key Property: Invalid Date Name - 'START	39005	
Invalid Input	Key Property: Invalid Date Code - 'DATE_1'	39006	
Invalid Input	The following key properties are required but do not have a value: 'LIST OF MISSING REQUIRED KEY PROPERTIES'	39007	
Invalid Input	Key Property: Invalid category value name for 'LOCATION' category: 'FAKE TOWN'	39008	
Invalid Input	Key Property: Invalid category value path for ''LOCATION category: 'FAKE/PATH/4U'	39009	
Invalid Input	Key Property: Invalid category value code for 'LOCATION' category: 'FAKE_CODE'	39010	
Invalid Input	Key Property: Invalid manager identifier entered for 'MANAGER' manager: 'FAKE.LOGIN'	39011	
Invalid Input	Key Property: Date format Key Property: Date format	39012	Expected format is YYYY-MM-DD

LaunchEvent Error Codes

<Type>	<Message>	<ErrorCode>	Description
Event Exists	*Named* event already exists for user: *ForWhom*	30000	Duplicate Event
Invalid Input	Failed to parse XML documents	30001	Caused by malformed XML
Invalid Input	Failed to add event to user as the sessionid executing the method is not an event coordinator	30002	Privilege error
Invalid Input	Unknown event name	30003	Undefined Event name
Invalid Input	Invalid category value name for *named category:value*	30004	The value in passed for <Type>Name</Type> did not match a valid category name
Invalid Input	Invalid category value code for *named* category:*value*	30005	The value in passed for <Type>Code</Type> did not match a valid category Code
Invalid Input	Invalid category value path for *named* category:*value*	30006	The value in passed for <Type>Path</Type> did not match a valid category Path
Invalid Input	Named category does not exist on the Event definition	30007	Launch data must map to field defined on the event definition.
Invalid Input	Named person does not exist on the Event definition	30008	Launch data must map to field defined on the event definition.
Invalid Input	The ForWhom_AuthParam used to identify a user is not used by any known user	30009	Invalid value external authorization parameter

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	The ForWhom_LoginID used to identify a user is not used by any known user	30010	Invalid value employee LoginID
Invalid Input	The ForWhom_Guid used to identify a user is not used by any known user	30011	Invalid value User GUID
Invalid Input	The ForWhom_Email used to identify a user is not used by any known user	30012	Invalid value email address
Invalid Input	The e-mail address used to identify a user is used by more than one user	30013	Email address must be unique if it is to be used as the primary identifier
Invalid Input	No user identification node could be found for the *nth* event node	30025	No valid ForWhom_* identifier was provided in the (specified) record
Invalid Input	Missing person (manager) name or Missing person (manager) value	30026	Person node included in record with no <Name> or <Value> node
Invalid Input	Missing EmployeeID Type for named person on the event	30056	Missing EmployeeIDType for a named person.
Invalid Input	Incorrect EmployeeIDType	31053	The EmployeeIDType {0} is unknown. Please use one of the following: LifeSuiteID, Employee_HRISID, Email, LoginID, GUID, or SSOAuthParam.
Invalid Input	Missing required information: *message text*	34000	A required event field specified in the message text has been omitted from the record.

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Invalid Category reference	36015	{0} cannot be used because it is not a leaf category value (i.e. It has child category values)

GetUserIDList Error messages

<Type>	<Message>	<ErrorCode>	Description
Error	Invalid input XML: {0}	42000	
Error	The synchronization token was provided cannot be used for this web service method. It may have been returned from a different web service and cannot be used with this web service, with the criteria specified.	42001	Invalid Synchronization token
Error	The synchronization token cannot be decrypted and appears to have been tampered with	42002	Invalid Synchronization token

GetUserProfileEx2 Error messages

<Type>	<Message>	<ErrorCode>	Description
UserError	The LoginId ({0}) used to identify a user is used by more than one user	31059	
UserError	No user could be found for the id: {0}	31060	

CategoryUpload Error Messages

<Type>	<Message>	<ErrorCode>	Description
	Failed to create restore point.	36000	

<Type>	<Message>	<ErrorCode>	Description
	'Code' node doesn't exist for <name of Category Value>	36001	
	Input structure doesn't match database structure for <Name of Category>	36002	
	Code is too long for {0}	36003	
	Malformed category upload XML	36004	
	Category Value with code '<Code>' doesn't exist.	36005	
	Code must be specified to retire Category Value '<Category Value name>'	36006	
	Invalid Portal Role: '<Eprise role>'	36007	
	'Update' and 'Replace' require all Category Value Codes to be populated and are not supported in your environment. Please use 'ReplaceAll' to bulk update your category values	36008	
	'<Category Name>': only one root Category Value is allowed per Category	36009	
	Internal Code '<Code>' already exists. Internal Codes must be unique if specified.	36010	
	A Category with the name '<Category Name>' already exists	36011	
	Category '<Category Name>' does not exist.	36012	

GetTasks Error messages

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Invalid category alias Code: 'NonExist'	30042	

<Type>	<Message>	<ErrorCode>	Description
Invalid Input	Invalid event status: 'zzz'	30062	
Invalid Input	Invalid category value code for *X* category: *Y value*	30005	
Invalid Input	Unknown task definition string ID: NonExistantTask	35002	
Invalid Input	Key Property: Invalid category alias Code: *X*	39002	
Invalid Input	Key Property: Invalid category value code for NonExistCatValue category: *X*	39010	
Invalid Input	Key Property: Invalid person Code: *X*	39004	
Invalid Input	The LoginID (NonExistantUser) used to identify a user is not used by any known user	39011	
Invalid Input	Invalid input XML: *description of invalid element*	42000	
Invalid Input	The synchronization token cannot be decrypted and appears to have been tampered with.	42002	

Appendix: Web services example processes

Launching event with rehires

Overview

This provides an outline of how to use the Onboarding SOAP APIs to launch events for new employees from an HRIS system.

Baseline – New hire events are launched from an HRIS integration

Web services used:

- BulkUserUpload

Process:

Valid user xml with an event node is sent to Onboarding via the BulkUserUpload web service. This user is created as a pending user, which is can be reviewed and approved in the Onboarding user interface.

Scenario 1 – HRIS maintains list of SilkRoad Onboarding users

Assumptions:

1. The HRIS system knows all the employees that have user accounts
2. Onboarding user accounts use a login ID or Authentication Parameter value that is 'known' to the HRIS system.
3. If the employee exists in Onboarding for a rehire, their user account will be 'retired'
4. All the event information for rehires is available in the HRIS system.

Web services used:

- BulkUserUpload
- XMLUserEdit
- LaunchEvent

Process:

Because the HRIS system is aware of which employees have accounts in Onboarding, the logic that creates employees and launches events in Onboarding branches prior to attempting to send data to Onboarding.

- If the user does not exist in Onboarding: Use BulkUserUpload to create a new user with an event (Baseline process above).
- If the user does exist in Onboarding:
    - Use XMLUserEdit to unretire the user account & update any Employee Profile field.
        - At this point the account is active and the user could log in.
        - No notification is sent to the user
    - Use LaunchEvent to supply the event information and launch the event for the active user.
        - All event information (categories, people and dates) must be supplied properly or the event launch will fail.
        - The event launched notification is sent; password cannot be sent in this notification for an active user.

Scenario 2 – HRIS does not maintain list of SilkRoad Onboarding users

Assumptions:

1. If the employee exists in Onboarding for a rehire, their user account will be retired

2. Onboarding user accounts use a login ID or Authentication Parameter value that is known to the HRIS system.
3. All the event information for rehires is available in the HRIS system.

Web services used:

- BulkUserUpload
- XMLUserEdit
- LaunchEvent

Process:

Since the HRIS system is not aware of which employees have accounts in Onboarding, it must attempt an action on a Onboarding user account before it knows is the employee has a user account.

- Use XMLUserEdit to attempt to unretire the user account that needs an event launched.
    - If user cannot be found; User does not exist:
        - Use BulkUserUpload to create a new user with an event (Baseline process above).
    - If user is found; and now unretired
        - Use XMLUserEdit to update any Employee Profile field.
            - At this point the account is active and the user could log in.
            - No notification is sent to the user
        - Use LaunchEvent to supply the event information and launch the event for the active user.
            - All Event information (categories, people and dates) must be supplied properly or the event launch will fail.
            - The event launched notification is sent; password cannot be sent in this notification for an active user.

Using SOAP API to get eForm Data

Integration with SilkRoad Onboarding

To programmatically retrieve eForm data from Onboarding, the client side integration must use a multi-step process using various web services published by Onboarding.
Background information:

Web Service documentation https://\[client\]Onboarding.silkroad.com/rc/RCHelp/Content/Additional_References.htm

In this example all the other forms will be retrieved when "Trigger Form" is completed. The "Trigger Form" will be the Federal W-4.

In the example below, we'll be using the GetCompletedFormList method.

1. Method: Login
    a. Send: User ID, Password
    b. Receive: Session ID
2. Method: GetCompletedFormList
    a. Send: Session ID, strXML
    b. <?xml version="1.0" encoding="UTF-8"?><GetCompletedFormsListInput><FormFilter> <IsFormDelivered>0</IsFormDelivered> <FormType>Federal_W_4</FormType> </FormFilter></GetCompletedFormsListInput>
        i. FormType: the type of the Trigger form, in this example: Federal_W_4
        ii. IsFormDelivered: 0=Not Delivered; 1 = Delivered
    c. Receive: XML string:
    d. <GetCompletedFormsListOutput><Forms><Form> <ObjectId>11628</ObjectId> <FormType>Federal_W_4</FormType> <Owner> <Guid>be6568a5-d74e-43b6-acf1a000f13e4b61</Guid> <Employee_HRISID /> <SSOAuthParam /> <LoginID>Demo.Onboarding</LoginID> <Email>ctino@silkroad.com</Email> <LifeSuiteID /> </Owner> <CompletedDate>2013-07-02</CompletedDate> <DeliveredDate xsi:nil="true" /> <Event> <EventID>56</EventID> <EventCode>Event_5</EventCode> <EventName>Demo Onboarding</EventName> </Event> </Form></Forms></GetCompletedFormsListOutput>
    e. For each Owner: GetCompletedFormList
        i. Send: Session ID, strXML <?xml version="1.0" encoding="UTF-8"?><GetCompletedFormsListInput> <FormFilter> <IsFormDelivered>0</IsFormDelivered> <FormOwnerFilter><EmployeeIDFilter> <EmployeeIDType>LoginID</EmployeeIDType> <ID>Demo.Onboarding</ID> </EmployeeIDFilter></FormOwnerFilter> </FormFilter></GetCompletedFormsListInput>
        ii. Receive XML sting with a Form node for each of the forms that that have been completed in that user's event.
3. For each Form ID
    a. GetFormXML
        i. Send: Session ID, ObjectID (Form ID)
        ii. Receive: Form data
4. Verify Data (Try HRIS import)
    a. If Import Successful
        i. MarkFormDelivered
            1. Send: Session ID, ObjectID (Form ID), 1
        ii. Receive: nothing
    b. If not successful
        i. Log problem/alert/do NOT mark trigger form Delivered
5. Logout
    a. Send: Session ID
    b. Receive: nothing

Index