RedCarpet
onboarding & life events

Web Service Guide

Version 2.9.0

SilkRoad technology
www.silkroad.com

SilkRoad RedCarpet WebService Guide
Document Version: July 2013
Product Version: RedCarpet v 2.9.0

Technical Support
SilkRoad technology, inc.
Web: http://support.silkroadtech.com
Phone: 866-329-3363

Headquarters
20 West Kinzie Street, Suite 1220
Chicago, IL 60654
1 (866) 329.3363 toll free (U.S. only)
+1 (336) 201.5100 phone
+1 (336) 201.5141 fax
www.silkroad.com

Table of Contents

RedCarpet Web Service

Overview

RedCarpet provides a web service for administrators to take advantage of RedCarpet features through an API. RedCarpet methods exist in these general categories of features:

1. An interface to retrieve RedCarpet user information
2. An interface to perform a user import and edit
3. An interface for LifeCycle events and forms.
4. An interface for Category Uploads.

Methods to Retrieve Employee Data

Four methods are available to retrieve employee data. One method returns a list of RedCarpet user ids and the other three accept a user id and return data about the specified user. The RedCarpet method to return a list of user ids is available to retrieve all the users in the system or just the users that have modified since the last retrieval.

Three RedCarpet methods are available to retrieve RedCarpet 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 user 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.

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

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

LifeCycle Event API

LifeCycle event methods are available to launch a 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 add and delete tasks from existing events are also available.

Forms API

A RedCarpet implementation includes the Eprise web service allowing programmatic access to retrieve and delete saved forms outside of the RedCarpet application. The available web methods include options to:

- retrieve completed forms based on various filter criteria
- retrieve PDF files (populated through RedCarpet)
- indicate whether a form has already been retrieved
- delete existing forms

Upload Category Data

RedCarpet category upload methods allows RedCarpet administrators to import and update category values and modify the hierarchy of the category structure. In addition to facilitating the integration with other applications the “Import and Export Category Management” feature eases maintenance of category value changes.

Prior to RedCarpet 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.

RedCarpet 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 specifications of the XML syntax can be found in the Category Upload Guide (“RCCategoryImport.pdf”)

Using RedCarpet Methods

Background on the RedCarpet Methods

RedCarpet uses an underlying content management engine called “Eprise” to: store the user data, handle the form templates and saved forms, and build and manage the integrated RedCarpet portals. The Eprise content management database handles all of RedCarpet’s content maintenance and security. This is transparent to our users. As an administrator using the RedCarpet API, you will use the Eprise web service to consume methods applicable to RedCarpet.

◁ The API references the naming convention of the underlying content management engine called “Eprise”.

Web Service URL

The web service is referenced with the following URL:

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

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

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

◁ Note: All of your RedCarpet pages are served over SSL.

While the majority of these methods do not apply to RedCarpet, you may use this page for verification of applicable methods and their parameters.

Logging in to Consume the Service

While the WSDL is public, consuming the RedCarpet 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 a RedCarpet employee with the correct privileges. It is important to note all RedCarpet methods are constrained by the same security (authentication and authorization) rules as the RedCarpet user interface (performing the same function).

◁ Note: RedCarpet role membership (and privilege assignment) may be set up by your RedCarpet systems administrator.

For additional information on

Method name: Login

Login to obtain a valid session id.

- Parameters

strLogInId	Valid Login ID of a privileged RedCarpet employee. The required employee privileges correlate to the privileges required for each method executed.	string
strPassword	Corresponding password	string

strRemoteIP	Optional. In most cases this parameter does not apply to RedCarpet methods. Populate this parameter only if this user is being authenticated by an external authentication server. The strRemoteIP would in this case, by populated with the ONE machine IP Address of authentication server ("remote browser").	string

- Returns

  sessionID(simple string) — valid Eprise session id

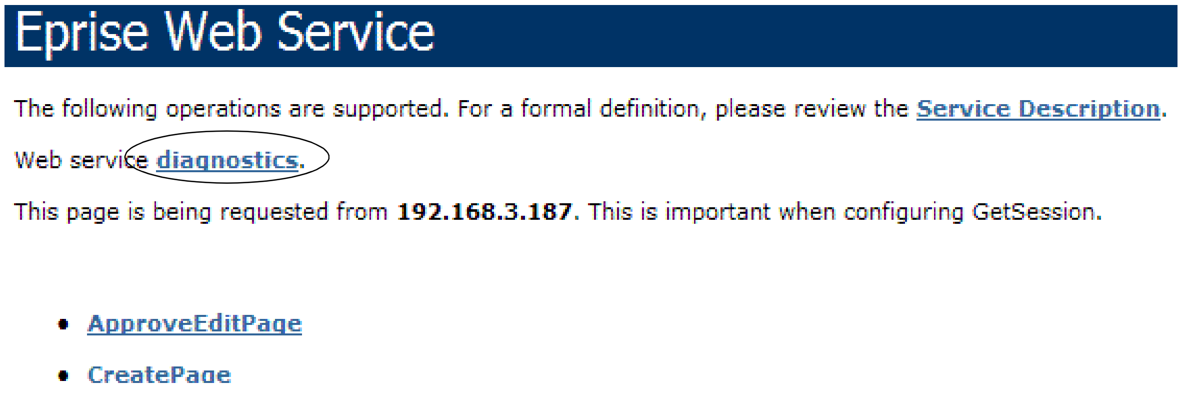
  ""(empty string) — unsuccessful login

- Code 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;

    }
```

Debugging Method Calls

The web service page includes a link labeled "diagnostics" to log the (input / output) activity specific to web service method calls. An administrator can use the the diagnostics page to (dynamically) turn on and off the debug logging feature as well a view diagnostic information in the log.


Eprise Web Service

The following operations are supported. For a formal definition, please review the Service Description.

Web service diagnostics.

This page is being requested from **192.168.3.187**. This is important when configuring GetSession.

- ApproveEditPage
- CreatePage

Steps to use debug logging

To access the method diagnostics:

1 Click the diagnostics link on the web service page (https:\\<your RC server>\eprise\WebServices)

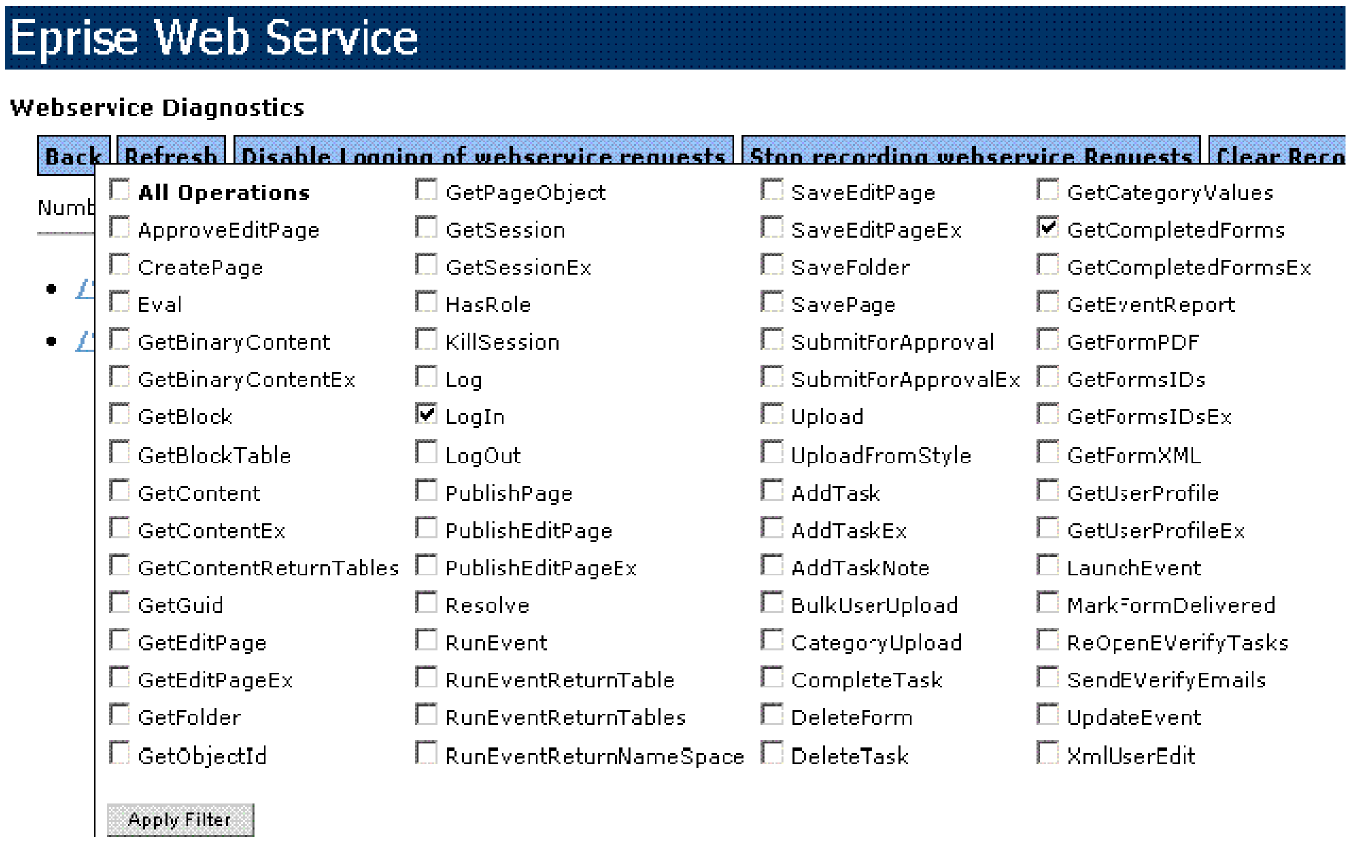
2 Click on one or both of the logging options:

- "Enable Logging of webservice requests" button to write the debug messages to the eprise log file (located on your RedCarpet server)
- "Start recording webservice Requests" button to display the debug messages on the diagnostics page

3 After enabling logging, the following options are available:

- Back button - click to return to the method overview page
- Refresh button - click to refresh the diagnostics page display
- Disenable Logging of webservice requests button - turn off the method logging to the eprise.log file
  - Disenable Logging is a toggle with the Enable logging option
- Stop recording webservice Requests button - turn off the method logging to the (current) page
  - Stop recording is a toggle with the Start recording option
- Clear Records button - clear the current page display
- Filter button - change the default setting ("All Operations") from logging all method calls to log only specific method calls.
  - Check the methods to log and/or view
  - Choose the "Apply Filter" button to apply the checked methods.

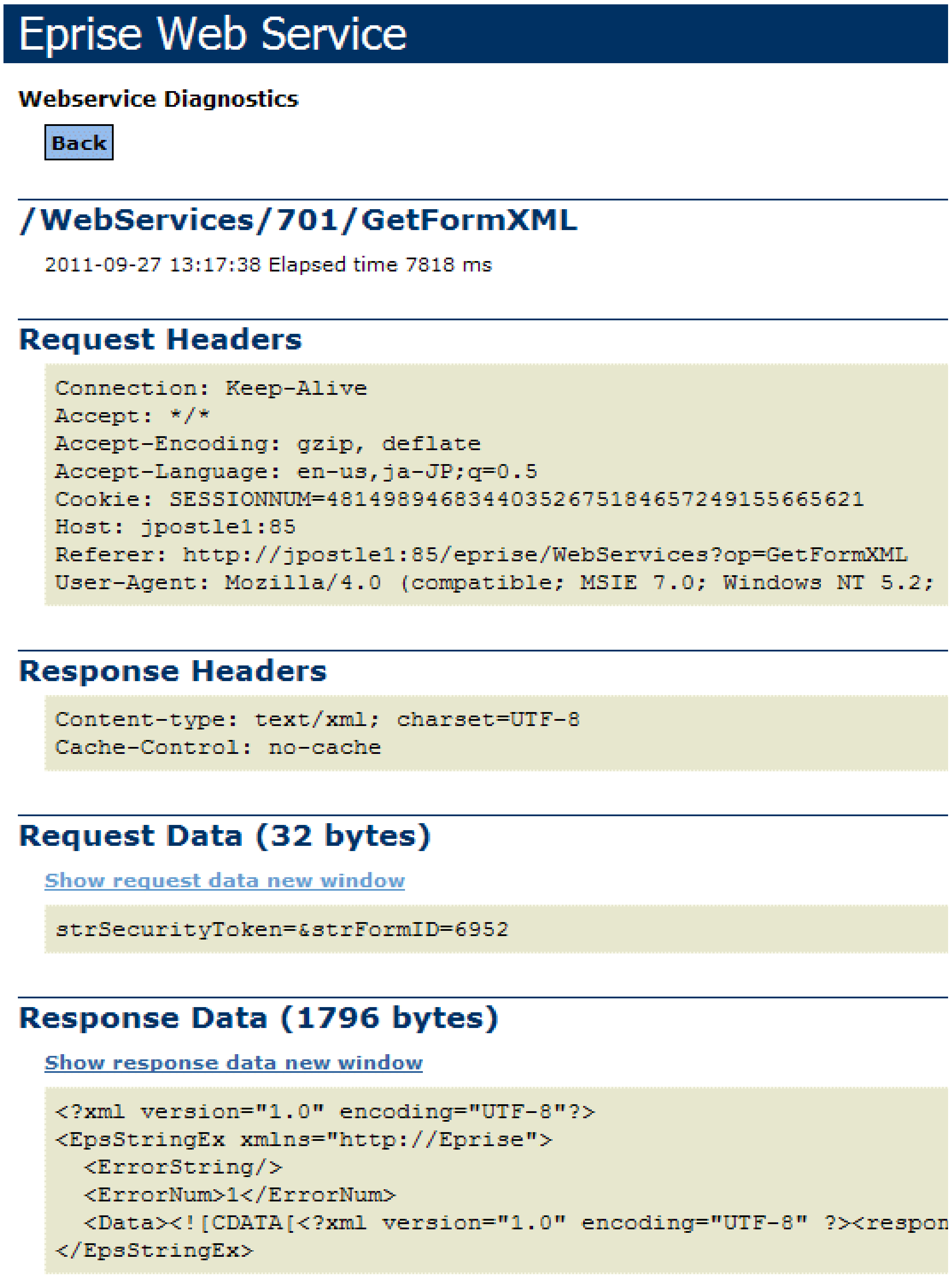
◁ Note: For security reasons, default configuration restricts the access to the diagnostics page to administrators. Contact SilkRoad support to discuss extending this to a non-administrator user.



☐ **All Operations**	☐ GetPageObject	☐ SaveEditPage	☐ GetCategoryValues
☐ ApproveEditPage	☐ GetSession	☐ SaveEditPageEx	☑ GetCompletedForms
☐ CreatePage	☐ GetSessionEx	☐ SaveFolder	☐ GetCompletedFormsEx
☐ Eval	☐ HasRole	☐ SavePage	☐ GetEventReport
☐ GetBinaryContent	☐ KillSession	☐ SubmitForApproval	☐ GetFormPDF
☐ GetBinaryContentEx	☐ Log	☐ SubmitForApprovalEx	☐ GetFormsIDs
☐ GetBlock	☑ LogIn	☐ Upload	☐ GetFormsIDsEx
☐ GetBlockTable	☐ LogOut	☐ UploadFromStyle	☐ GetFormXML
☐ GetContent	☐ PublishPage	☐ AddTask	☐ GetUserProfile
☐ GetContentEx	☐ PublishEditPage	☐ AddTaskEx	☐ GetUserProfileEx
☐ GetContentReturnTables	☐ PublishEditPageEx	☐ AddTaskNote	☐ LaunchEvent
☐ GetGuid	☐ Resolve	☐ BulkUserUpload	☐ MarkFormDelivered
☐ GetEditPage	☐ RunEvent	☐ CategoryUpload	☐ ReOpenEVerifyTasks
☐ GetEditPageEx	☐ RunEventReturnTable	☐ CompleteTask	☐ SendEVerifyEmails
☐ GetFolder	☐ RunEventReturnTables	☐ DeleteForm	☐ UpdateEvent
☐ GetObjectId	☐ RunEventReturnNameSpace	☐ DeleteTask	☐ XmlUserEdit

- Note: the filter option applies to both the log file entries and the page display (“recording”).

The diagnostic output includes the method executed, request header data, response headers, the input parameters, and the method output.



Eprise Web Service

Webservice Diagnostics

Back

/WebServices/701/GetFormXML

2011-09-27 13:17:38 Elapsed time 7818 ms

Request Headers

```
Connection: Keep-Alive
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us,ja-JP;q=0.5
Cookie: SESSIONNUM=4814989468344035267518465724915566562l
Host: jpostle1:85
Referer: http://jpostle1:85/eprise/WebServices?op=GetFormXML
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2;
```

Response Headers

```
Content-type: text/xml; charset=UTF-8
Cache-Control: no-cache
```

Request Data (32 bytes)

Show request data new window

```
strSecurityToken=&strFormID=6952
```

Response Data (1796 bytes)

Show response data new window

```
<?xml version="1.0" encoding="UTF-8"?>
<EpsStringEx xmlns="http://Eprise">
  <ErrorString/>
  <ErrorNum>1</ErrorNum>
  <Data><![CDATA[<?xml version="1.0" encoding="UTF-8" ?><respon
</EpsStringEx>
```

Chapter 1: RedCarpet Retrive User Data API

Chapter 1

Retrieve User Attributes

Note: This guide uses the terms “user” and “employee” synonymously.

The following methods are available to retrieve employee attributes:

1 GetUserIDsList 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 specic user attributes. GetUserIDsList 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.

2 GetUserProfileEx and GetUserProfileEx2 return all attributes associated with any active RedCarpet user. GetUserProfileEx2 includes additional fields including addtional E-Verify case data, and two additional user id identifiers. Both methods output an xml document with name\value pairs for all attributes.

3 GetUserProfile returns an array of attributes associated with any active RedCarpet 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. unique form ids.

Employee ID Types

Users are uniquely identified in RedCarpet based on your implementation.

You may query user data based on the appropriate RedCarpet user for your implementation. All of these options are returned by GetUserIdsList. The following criteria are available to be used to identify a user:

- LoginID - required for all users. LoginID is required to be unique in RedCarpet.
- Email - require for all users. Email is required to be unique based on a configuration setting.
- Guid - a (RedCarpet) generated system key. Guid is unique for RedCarpet users. It is not exposed through the RedCarpet user interface.
- LifeSuiteID - a (SilkRoad Technology) generated system key. This is unique for all LifeSuite users. It is not exposed through the RedCarpet user interface. This key exists for integrating user data across the LifeSuite applications. The LifeSuiteID can be used as an input parameter with GetUserProfileEx2, BulkUserUpload, XMLUserEdit, and GetUploadedDocumentList.
- Employee_HRISID - optionally configured for collection through the RedCarpet interface. Employee_HRISID is optionally configured to be required. It is unique for RedCarpet users. This field exists to cross reference an employee with their ID in another (external corporate) system. It's label and value is configured and populated based on customer specific requirements. SilkRoad recommends you populate this value for system cross reference.The Employee_HRISID can be used as an input parameter with GetUserProfileEx2, BulkUserUpload, XMLUserEdit, and GetUploadedDocumentList.
- SSOAuthParam - optionally configured and referred to as “external authentication”. The authentication parameter is the user id used to authenticate in an externalsystem. RedCarpet uses this parameter when it is configured for “single sign on”. Customers may choose to populate the SSOAuthParam with the external system identifier even if they are not using external authentication. The SSOAuthParam can be used as an input parameter with GetUserProfileEx2, BulkUserUpload, XMLUserEdit, and GetUploadedDocumentList. For backward compatibility BulkUserUpload and XMLUserEdit also accept “AuthParam”

GetUserIDList

Usage

GetUserIDList returns a list of user id’s. 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 value. 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 RedCarpet call GetUserIDList passing a null for the synchronization token. RedCarpet will return a 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, you must call GetUserIDList using the last sychronization token return in the previous results set.

When using this technique to synchronize data from RedCarpet 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)
- E-Verify Case data (

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 ID’s could be passed to GetUserProfileEx2 to obtain the employee information for each employee who has been updated since the last call.

Input for GetUserIDList

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strXML	Required XML is described below. The XML specifies your sychronization 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>
```

Nodes in GetUserIDList

Node	Description	Notes
`<?xml version="1.0" encoding="utf-8"?>`	XML Document header	Required
`<GetUserIDListInput>`	Parent node	
`<SynchronizationToken />`	Synchronization token marker for where to begin the results set. Null returns all user records in update order. The most recent update will be the last in the results set. A valid sychronization 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 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 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 the employees that meet the sychronization criteria. The nodes in the return data are as follows:

<?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 is a unique RedCarpet system value returned for each user
<Employee_HRISID/>	Employee_HRISID returned if populated through RC. Optionally configured as required.
<SSOAuthParam />	SSOAuthParam returned if populated through RC. Configured if authentication type is external configuration. Optionally configured if standard authentication.
<LoginID></LoginID>	LoginID is required and unique in RedCarpet
<Email></Email>	Email is required and optionally unique.
<LifeSuiteID />	LifeSuiteID is a unigue LifeSuite system populated value returned if it has been assigned by a LifeSuite application
<EmployeeStatus>Active</ EmployeeStatus>	EmployeeStatus returns Active or Retired for each user.
<SynchronizationToken> SynchronizationToken>	Sychronization 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
</Users>	End Users list. Occurs once for each document

GetUserIDList Error Messages

<Type> Value	<Message>Value	<ErrorCode> Value	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

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>")
```

Sample 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
       +3KfH87DN3GNxFKWPDurca47G1jt1uqNOUx470Q==</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 the event data including event named people, categories and dates, and e-verify case data if applicable.

Parameters for GetUserProfileEx2

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strXML	Required 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 in GetUserProfileEx2

Node	Description	Notes
`<?xml version="1.0" encoding="utf-8"?>`	XML Document header	Required
`<GetUserProfileEx2Input>`	Parent node	Required
`<EmployeeIDFilter>`	Parent node	Required

Node	Description	Notes
`<EmployeeIDType></EmployeeIDType>`	ID type	Required. Accetable values are: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteID See "Employee ID Types" on page 8 for description. example: `<EmployeeIDType>LoginID</EmployeeIDType>`
`<ID></ID>`	ID value	Required. 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	Required
`</GetUserProfileEx2Input>`	close parent node	Required

Returns

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 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 including 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> Value	**<Message>Value**	**<ErrorCode> Value**
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

Sample Output

```
<EpsStringEx xmlns="http://Eprise">
  <ErrorString>No Error</ErrorString>
  <ErrorNum>1</ErrorNum>
  <Data>
  <![CDATA[ <?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]]  ]]><![CDATA[ ></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>
```

```
      <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

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strLoginID	Required if strFKUserID is not populated. RedCarpet 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.

The structure is:

```
EpsTableEx{

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.

““ or “*xxxx*” — return code description 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.

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","")
```

Sample 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, a drop down list selection of Interests, and a comma separated multi-select called “Days_avail”.

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>
<LoginID>hwise</LoginID>
  <AuthParam />
  <Email>hwise@work.com</Email>
- <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 for GetUserProfile

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strLoginID	Required if strFKUserID is not populated. RedCarpet 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.

The structure is:

```
EpsTableEx{

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.

"" or "*xxxx*" — return code description 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.

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","")
```

Sample 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><![CDATA[example.user@silkroadtech.com]]></string>

      <string/>

      <string><![CDATA[Include name, phone, email address if applicable,]]></string>

    </ArrayOfString>

  </Data>
```

Chapter 2: 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.

The XML syntax for defining employees is documented in a separate document called the User Import Guide (named "RCBulkUserImportandEdit.pdf"). The User Import Guide describes the syntax and all of the return codes for the bulk import and edit features as well as the processing of pending users through the RedCarpet interface.

Import New Users

An XML record used to create both pending and active users.

BulkUserUpload

Usage

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

Parameters

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

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.

““ 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 import new RedCarpet 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>
Upload finished
```

Bulk Import Error Codes

Error codes are returned in the same format for the post and the method call. See the User Import Guide for error code information returned in the Data string.

Edit Existing Users

An XML record used to update active users.

XMLUserEdit

Usage

Used to update one our more existing active users using an XML formatted file.

Parameters

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

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.

““ 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 RedCarpet 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>
Upload finished
```

XMLUserEdit Error Codes

Error codes are returned in the same format for the post and the method call. See the “User Import Guide” for error code information returned in the data string.

Chapter 3: Event and Task APIs

Event and task related APIs are available to create and edit employee events programmatically. 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 Red Carpet interface).

Events APIs

Two methods are available to manage employee events

- launch a new event for an existing employee
- edit an existing LifeCycle event

The Event definition must be defined through the RedCarpet 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 (launch a new event and update an existing event) use two arguments.

1 A Session Id as described in “Logging in to Consume the Service” on page 3 of this document.

- The RedCarpet user executing the Events methods must be a member of the Event Coordinator team for each event included in the Events record.

◁Note: SessionID for event methods must include Event Coordinator membership.

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.

Field Descriptions of the xmlEventData

◁Note: See below for additional information on tags for events.

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:

1The child element must contain one of the unique user identifiers described below.

2Child 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: an element to uniquely identify an employee to benefit from the event or the event update.

You may one of choose 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.

◁Note: SilkRoad recommends using the ForWhom_AuthParam

"ForWhom" Identifier	Description
`ForWhom_LoginId`	RedCarpet 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 unique.
`ForWhom_Guid`	RedCarpet creates and stores a GUID for each employee. This value is returned from GetUserProfileEx method.

2 You may use this flexible technique allowing your to set the identifier type by specifying it as a value. Note: f you want to identify the user by the LifeSuiteID or the EmployeeHRISID your must use either this technique or the next.

```
<ForWhom_EmployeeID>

    <EmployeeIDType></EmployeeIDType>

    <ID></ID>

</ForWhom_EmployeeID>
```

where EmployeeIDType is on of the values described in See “Employee ID Types” on page 8.

3 LaunchEvent and UpdateEvent also accept this syntax.

```
<EmployeeID>

    <EmployeeIDType></EmployeeIDType>

    <ID></ID>

<EmployeeID>
```

where EmployeeIDType is on of the values described in See “Employee ID Types” on page 8.

Part Two: The named event to be launched or updated it’s corresponding event field(s)

The naming convention of the subchild element tags is determined by the fields defined in the event definition (in the RedCarpet user interface).

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

When creating your event syntax a LifeCycle event note that Event categories are populated with a **single 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.

A description of event field names is as follows:

Element Syntax	**Required Status if an event is included**	**Description**
Category	Yes	Occurs for each category defined on the event Example syntax: <Category>
Name	Yes	Occurs once as a child element of <Category> for each category element defined on the event Valid values are text as it is entered in the “Alias Name” field of the Manage Events page. Example syntax: <Name>Location</Name>
Value	Yes	Occurs once as a child element of <Category> for each category element defined on the event 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>North East</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>

Element Syntax	Required Status if an event is included	Description
Type	Yes	Occurs once as a child element of <Category> for each category element defined on the event 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> specifies the 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. SilkRoad recommends the use of the Code type to promote a maintainable process in case category values are renamed or moved. <Type>Name</Type> **can not be used if duplicate values exist** within your category tree.I

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.

Non-category field syntax:

Element Syntax	Required Status if an event is included	Description
Person	Yes	<Person> is a parent element. Occurs one time for each named person.on the event.
Name	Yes	Child element of <Event> Valid values are text as it is entered in the "Person Name" field of the Manage Events page. Example: <Name>HR Coordinator</Name>
Value	Yes	Child element of <Event> Valid value is a specific person who is a member of the specified (Person Name) Team on the Manage Events page. The specific person can be identified by one of these unique identifiers: • email address* • user GUID • login ID • External Authentication ID *If email address is not unique it is not a valid identifier Example: <Value>George.Rogers</Value>
Date	Yes	<Date> is a parent element. Occurs one time for each "date name" defined on the event.
Name	Yes	Child element of <Date> Valid values are text as it is entered in the "Date Name" field of the Manage Events page. Example: <Name>Start</Name>
Value	Yes	Child element of <Date> Date value to corresponding <Name> Expected format is YYYY-MM-DD

Output Specifications

Similar to the BulkImport and XMLEditUser methods, each method returns a table of type EpsStringEx.

The structure is:

```
EpsStringEx{

   String ErrorString;

   int ErrorNum;

   String Data;

   }
```

ErrorString and Error Num

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 is equal to “There were errors”. Errors 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.

Sample xmlEventData

A sample of Events record used in the Events methods is as follows

LaunchEvent Method

```
<?xml version="1.0" encoding="UTF-8"?>
<Events>
   <Event>
      <ForWhom_EmployeeID>
         <EmployeeIDType>LoginID</EmployeeIDType>
         <ID>James.Develin</ID>
      </ForWhom_EmployeeID>
      <Name>Onboarding-EVerify</Name>
      <Person>
         <Name>Onboarding Coordinator</Name>
         <Value>Amy.HrCoord@mySRT.com</Value>
      </Person>
      <Person>
         <Name>Manager</Name>
         <Value>Robert.Manager@MySRT.com</Value>
      </Person>
      <Date>
         <Name>Start</Name>
         <Value>2012-12-12</Value>
      </Date>
      <Category>
         <Name>Department</Name>
         <Value>Customer Service</Value>
         <Type>Name</Type>
      </Category>
      <Category>
         <Name>Location</Name>
         <Value>Boston</Value>
         <Type>Name</Type>
      </Category>
   </Event>
</Events>
```

Usage

Used to programmatically launch an event for an existing employee.

Code examples

To launch a LifeCycle event an XML record.

```
string sEvents = ‘<?xml version="1.0"?><Events>’ +
"  <Event>" +
" <ForWhom_EmployeeID>" +
  “ <EmployeeIDType>LoginID</EmployeeIDType>" +
  “ <ID>James.Develin</ID>" +
  “ </ForWhom_EmployeeID>" +
"  <Name>Onboarding</Name> " +
"  <Person>" +
"    <Name>HR Coordinator</Name> " +
"    <Value>hwise@redcarpet.com</Value> " +
"  </Person>" +
"  <Person>" +
"    <Name>Manager</Name>" +
"    <Value>mbloom@redcarpet.com</Value>" +
"  </Person>" +
"  <Date>" +
"    <Name>Start</Name>" +
"    <Value>2008-11-13</Value>" +
"  </Date>" +
"  <Category>" +
"    <Name>Department</Name>" +
"    <Value>MRKNG</Value>" +
"    <Type>Code</Type>" +
"  </Category>" +
"  <Category>" +
"    <Name>Job Type</Name> " +
"    <Value>810SALES</Value>" +
"    <Type>Code</Type>" +
"  </Category>" +
"  <Category>" +
"    <Name>Location</Name>" +
"    <Value>NE_FRA</Value>" +
"    <Type>Code</Type>" +
"  </Category>" +
" </Event>" +
"</Events>"”
Eprise.EpsStringEx ret = service.LaunchEvent(sessionnum,sEvents);
if (ret.ErrorNum != 1)
{
    Console.Out.WriteLine("Error while launching events" + " Error:" + ret.Data);
    return;
    }
```

Parameters

strSecurityToken	Required. Valid Session ID for consuming service	simple string
xmlEventData	Required. XML record formatted following the rules outlined in "Field Descriptions of the xmlEventData" on page 38.	simple string

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

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.

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:

1 (one) — method succeeded

0 (zero) — method failed

Data

A record will be returned with descriptive XML

The elements of the record returned in the data string follows

Sample output for the data string for 2.8.0 and higher:

Element Syntax	Value	Description
LaunchEventResults	Parent node.	
Note	X event created	Successful eent 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 identifer 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>

lSample output for LaunchEvent:

```
<?xml version="1.0" encoding="UTF-8"?>
<LaunchEventResults>
<Note>1 events launched</Note>
<LaunchedEvents>
<LaunchedEvent>
<ForWhomEmployee>
<Guid>fb08d035-a436-4f05-85237a07073a2848</Guid>
<Employee_HRISID>3008</Employee_HRISID>
<SSOAuthParam />
<LoginID>Nate.Ebner</LoginID>
<Email>Ebner@mysrt.com</Email>
```

```
<LifeSuiteID />
</ForWhomEmployee>
<Name>Onboarding-EVerify</Name>
<EventID>44</EventID>
</LaunchedEvent>
</LaunchedEvents>
<Note>0 events not launched</Note>
<Note>Parsing started at Friday, December 14, 2012 12:11:40 PM and
ended at Friday, December 14, 2012 12:11:41 PM</Note> </
LaunchEventResults>
```

Note: by default the LaunchEvent output is transformed to 2.7.x output. In order to return output in this format you must disassociate the LaunchEvent operation XLST. See see “Default Configuration” on page 121 for additional information.

LaunchEvent Error Codes

<Type> Value	<Message>Value	<ErrorCode> Value	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 ecause 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 defintion.

<Type> Value	<Message>Value	<ErrorCode> Value	Description
Invalid Input	Named person does not exist on the Event definition	30008	Launch data must map to field defined on the event defintion.
Invalid Input	The ForWhom_AuthParam used to identify a user is not used by any known user	30009	Invalid value external auth paramethers
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 addresss 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 persond.
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> Value	<Message>Value	<ErrorCode> Value	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)

UpdateEvent Method

Usage

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

Code examples

To launch a LifeCycle event an XML record.

```
string sEvents = ‘<?xml version="1.0"?><Events>’ +
" <Event>" +
"  <ForWhom_LoginID>nwhite213</ForWhom_LoginID>" +
"  <Name>Onboarding</Name> " +
"  <Person>" +
"    <Name>HR Coordinator</Name> " +
"    <Value>lsmith@redcarpet.com</Value> " +
"  </Person>" +
‟ </Event>" +
"</Events>"”
Eprise.EpsStringEx ret = service.UpdateEvent(sessionnum,sEvents);
if (ret.ErrorNum != 1)
{
    Console.Out.WriteLine("Error while updating events" + " Error:" +
    t.ErrorString);
    return;
    }
```

Parameters

strSecurityToken	Required. Valid Session ID for consuming service	simple string
xmlEventData	Required. XML record formated following the rules outlined in "Field Descriptions of the xmlEventData" on page 38	simple string

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

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

Any correctly privileged user (Event Coordinators) can manually add task to existing events through the RedCarpet user interface, by using the "Add a Task" button on the Employee Task List page.

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:

1 (one) — method succeeded

0 (zero) — method failed

Data

An XML record will be returned.

An example return string is as follows:

A sample return is a follows:

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

<EpsStringEx xmlns="http://Eprise">

   <ErrorString>No Error</ErrorString>

   <ErrorNum>1</ErrorNum>

       <Data><![CDATA[<UpdateEventResults>
    <Note>1 events updated</Note>
    <UpdatedEvents>
    <UpdatedEvent>
    <ForWhomEmployee><Guid>78477262-a948-4ce4-bfc88a5b20f5ed7b</Guid>
    <Employee_HRISID>3007</Employee_HRISID>
    <SSOAuthParam></SSOAuthParam>
    <LoginID>James.Develin</LoginID>
    <Email>Develin@mysrt.com</Email>
    <LifeSuiteID></LifeSuiteID></ForWhomEmployee>
    <Name>Onboarding-EVerify</Name>
    <EventID>42</EventID>
    </UpdatedEvent></UpdatedEvents>
    <Note>0 events not updated</Note><Note>Parsing started at Monday, December 17,
    2012 4:10:15 PM and ended at Monday, December 17, 2012 4:10:17 PM</Note>
    </UpdateEventResults>]]>
    </Data>

</EpsStringEx>
```

Note: by default the UpdateEvent output is transformed to 2.7.x output. In order to return output in this format you must disassociate the LaunchEvent operation XLST. See see “Default Configuration” on page 121 for additional information.

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 eent 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 identifer 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>

<Type> Value	<Message>Value	<ErrorCode> Value	Description
Invalid Input	Failed to parse XML documents	30001	Caused by malformed XML
Invalid Input	Failed to update event to user ecause the sessionid executing the method is not an event coordinator	30020	Privilege error
Invalid Input	Unknown event name	30003	Undefined Event name

<Type> Value	**<Message>Value**	**<ErrorCode> Value**	**Description**
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 defintion.
Invalid Input	Named person does not exist on the Event definition	30008	Update data must map to field defined on the event defintion.
Invalid Input	The ForWhom_AuthParam used to identify a user is not used by any known user	30009	Invalid value external auth paramethers
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 addresss 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

<Type> Value	<Message>Value	<ErrorCode> Value	Description
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 persond.
Invalid Input	ForWhom_x identifier node cannot be blank	31044	The ForWhom_x identifierspecified 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.
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

Two methods are available to add and delete tasks from existing employee events:

- Add a new task for an existing LifeCycle event for an employee
- Delete an existing task from an existing LifeCycle event

In order to add or delete a task, the Task definition must be defined through the RedCarpet user interface. The “Task Definition ID” field defined in the task definition is used as the unique identifier of the task.

Input Specifications

Both of the task methods (add a new task and delete an existing task) use similar arguments:

◁Note: SessionID for event methods must include Event Coordinator membership.

1 strSecurityToken - A Session Id as described in “Logging in to Consume the Service” on page 3 of this document.

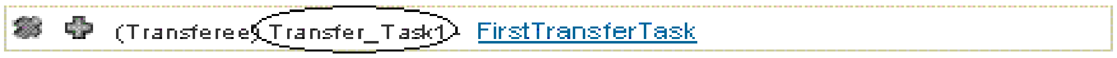
- The RedCarpet user executing the methods must be a member of the Event Coordinator team for each event included in the strEventName parameter OR
- The RedCarpet user executing the methods must be a member of a team with ViewAllTasks permission

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 will also accept 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 (Administration -> Manage Events - Manage Events \ Manage Task Definitions \ 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 “Transfer_Task1”


(Transferee(Transfer_Task1) FirstTransferTask

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 will prevent the task from being deleted. A value of False will allow a task with an associated form to be deleted.

Output Specifications

Each method returns a table of type EpsStringEx.

The structure is:

```
EpsStringEx{

   String ErrorString;

   int ErrorNum;

   String Data;
}
```

ErrorString and Error Num

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 is equal to “There were errors”. Errors 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 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.

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 as follows:.

```
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 Specifications section).	simple string

strForWhomUser	Required. The value refers to the RedCarpet 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 Specifications 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

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:

1 (one) — method succeeded

0 (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>
  <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	RedCarpet 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 session id privilege criteria.

AddTaskEx Method

Usage

Used to programmatically add a dependent task on an existing event for an existing employee and opting to add the task as either an unassociated task or a dependent task.

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 AddTaskEx method instead of the AddTask method.

Both the AddTask and AddTaskEx methods can handle adding dependent tasks if the parent task is defined as “Always Reopen Dependent Tasks”.

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 as follows:.

```
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 Specifications section).	simple string
strForWhomUser	Required. The value refers to the RedCarpet 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 Specifications 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 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..	simple string

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

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:

1 (one) — method succeeded

0 (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>
  <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 AddTaskEx:

<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	RedCarpet 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 session id privilege criteria.

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 Specifications 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 Specifications 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

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:

1 (one) — method succeeded

0 (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	RedCarpet 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

<Type> Value	<Message>Value	<ErrorCode> Value	Description
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 session id privilege criteria.

Chapter 4: Forms and Documents API

Forms and Documents APIs

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

- retrieve saved forms and completed forms based on various filter criteria
- retrieve PDF files (populated through RedCarpet)
- indicate if a form has already been retrieved
- delete existing forms

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

- retrieve uploaded documents accessible from the Edit Employee tab pages

Retrieve Existing Forms

Four web methods are available to retrieve existing forms.

Either method can be used to retrieve a list of form id's that meet specific criteria.

The four methods are very similar to each other with an important distinction of the date criteria.

- GetFormsIDs and GetFormsIDsEx use the *form creation date*
- GetCompletedForms and GetCompletedFormsEx uses the *form completion date*
- GetCompletedFormsList includes key properties and employee status.

All forms methods allow you to query for forms based on any combination of the following criteria:

    - when a form was completed
    - who the form was completed on behalf of
    - the type of form (form template name)

- if the form has previously been retrieved
- the verification status of a form

GetFormsIdsEx and GetCompletedFormsIdsEx methods allow you to query based on:

- what event the form was associated with
- what event category values apply to the form benefiter

GetCompletedFormsList method allow you to query based on:

- Employee Key Properties
- Employee Status (Active or Retired)

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 method to retrieve the form data.

◁ GetFormsIDsEx and GetCompleted FormsIDsEx include EventName and Category values in the selection criteria.

GetFormsIDs and GetCompletedForms

Usage and Parameters of GetFormsIDs

Retrieves a list of (saved) form id's that meet the specified criteria.

Parameters for GetFormsIDs

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strStartingDate	Optional. 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	Optional. 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	Optional. RedCarpet 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	Optional. 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	Optional. Specify a single RedCarpet Form name.	simple string
strDelivered	Optional. 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	Optional. 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

Usage and Parameters of GetCompletedForms

Retrieves a list of completed form id's that meet the specified criteria.

Parameters for GetCompletedForms

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strStartingDate	Optional. 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 id's greater than the strStartingDate	simple string
strEndingDate	Optional. 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 id's less than the strEndingDate	simple string
strLoginID	Optional. RedCarpet 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
strFormType	Optional. Specify a single RedCarpet Form name.	simple string
strDelivered	Optional. 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

string strEventName	Optional. 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 RedCarpet user interface.	simple string

Input for GetCompletedFormsList

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strXML	Required XML is described below. The XML specifies the document criteria for selection.	simple string

Nodes in GetCompletedFormsList strXML

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

`<CompletionDateRange>`
`<BeginDate>`
`</BeginDate>`
`<EndDate>`
`</EndDate>`
`</CompletionDateRange>`
	Optional - date range of upload
	
	Occurs once.
		Begin Date and / or End Date can be specified.
		
		Fomat is YYYY-MM-DD

Node	Description	Notes
<FormOwnerFilter> <EmployeeIDFilter> <EmployeeIDType> </EmployeeIDType> <ID> </ID> </EmployeeIDFilter> </FormOwnerFilter>	Optional EmployeeIDType and ID occurs for as many Employee ID as to be returnd, (see example below)	Possible values for EmployeeIDType are: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteI
<KeyPropertyFilter> <CategoryValue> <CategoryCode/> <CategoryValueCode /> <Person> <PersonCode> <EmployeeID> <EmployeeIDType> </EmployeeIDType> <ID> </ID> </PersonCode> </Person> <Date> <DateCode> <DateRange> <BeginDate/> <EndDate/> </DateRange> </DateCode>	Optional <KeyPropertyFilter> Occurs once per request. Child nodes occur multiple time as applicable to the employee definition.	Possible values for EmployeeIDType are: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteI Date format is YYYY-MM-DD DateCode and PersonCode are specified as they are defined in Key Properties.

Node	Description	Notes
<EmployeeStatusFilte> </EmployeeStatusFilte	Optional	Expected Values: Active Retired
<FormTypeFilter> <FormType> </FormType> </FormTypeFilter/>	Optional. FormType can occur many times per call	FormType values match the Form name defined in RedCarpet.
<EventFilter> <EventID> </EventID> <EventCode> </EventCode> <EventName> </EventName> <CategoryValue> <CategoryCode> </ CategoryCode> <CategoryValueCode></ CategoryValueCode> </CategoryValue> </EventFilter>	Optional	
</GetCompletedFormsListInput>	close parent node	

Sample strXML

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>2012-11-15</BeginDate>

       <EndDate>2012-12-15</EndDate>

    </CompletionDateRange>

  </FormFilter>
```

```
</GetCompletedFormsListInput>
```

Returns

An XML document

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 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.

FormID is in a column called "ObjectId"

◁ Note: See also - Sample Code Example at the end of this document.

Code examples

To return all existing forms:

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

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

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

To return all completed existing "I-9" forms:

```
Eprise.EpsTableEx t = service.GetCompletedForms(sessionnum,"","","","I-9","","")
```

Sample Return Data

An example return is as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<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>5684</string>
      <string>Parking_Assignement_Form</string>
      <string>1f80999e-d539-4fca-9e1a9ce864275023</string>
      <string>2009-06-18</string>
      <string/>
      <string>Onboarding</string>
      <string>49</string>
    </ArrayOfString>
    <ArrayOfString>
      <string>5685</string>
      <string>Form_I_9_2009</string>
      <string>1f80999e-d539-4fca-9e1a9ce864275023</string>
      <string/>
      <string/>
      <string>Onboarding</string>
      <string>49</string>
    </ArrayOfString>
    <ArrayOfString>
      <string>5686</string>
      <string>Confidentiality_Agreement</string>
      <string>1f80999e-d539-4fca-9e1a9ce864275023</string>
      <string/>
      <string/>
      <string>Onboarding</string>
      <string>49</string>
    </ArrayOfString>
  </Data>
</EpsTableEx>
```

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[

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

<GetCompletedFormsListOutput xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">

   <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>

         </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>
```

GetFormsIDsEx and GetCompletedFormsIDsEx

Usage

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

- Use GetFormsIDsEx to retrieve saved Form IDs by creation date
- Use GetCompletedFormsIDsEx to retrieve completed Form IDs by completion date

Syntax of the Category Parameter

GetFormsIDsEx and GetCompletedFormsIDsEx allow for an optional input parameter of category values as a filter criteria for the results set. Categories defined in your implementation of RedCarpet are customized for your company. The values are determined by your implementation.

Categories have multiple uses in RedCarpet. In the context of forms, you may filter on event categories to retrieve specific categories of forms. The event category values selected for the launched event will be 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 2 elements:

1 Alias (Category) name. The string should be specified as defined in the Manage Event page visible in the RedCarpet user interface.

2 Unique (category) value. The string should be specified as defined in the Manage Hierarchical Category Values page in the in RedCarpet user interface. The unique value can be specified in one of three formats.

   - by name
   - by internal code
   - 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
```

Parameters for GetFormsIDsEx

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strStartingDate	Optional. 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	Optional. 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	Optional. RedCarpet 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	Optional. 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	Optional. Specify a single RedCarpet Form name.	simple string
strDelivered	Optional. 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	Optional. 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

strEventName	Optional. 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 RedCarpet user interface.	simple string
arrCategoryFilter	Optional. 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 RedCarpet user interface. See "Syntax of the Category Parameter" on page 81	ArrayOfArrayOfString

Parameters for GetCompletedFormsIDsEx

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strStartingDate	Optional. 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	Optional. 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	Optional. RedCarpet 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	Optional. 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	Optional. Specify a single RedCarpet Form name.	simple string
strDelivered	Optional. 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	Optional. 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 RedCarpet user interface.	simple string
arrCategoryFilter	Optional. 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 RedCarpet user interface. See “Syntax of the Category Parameter” on page 81	ArrayOfArrayOfString

Returns

A table of type EpsTableEx.

The structure is:

```
EpsTableEx{

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.

"" or "*xxxx*" — return code description 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.

Sample output is as follows:

```
<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>
```

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.GetFormsIDsEx(sessionnum, "", "", "", "", "", "", "",
arrCats);
```

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

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

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

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

GetFormXML

Usage

Retrieves an existing XML form based on the specified FormID.

Parameters for GetFormXML

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strFormID	Required. A single formID . FormID is returned by GetFormsIDs, GetFormsIDsEX or GetCompletedFormsIDEx method as <ObjectID>	simple string

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.

““ 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

Sample output of the Data string (“`ret.Data`”) `for` a form named `TEST_W4 with five fields named SSN, City, State, Zip, and DT_NOW is as follows:`

```
<?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>
```

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”;
```

Mark a Form as Delivered

A delivery status flag is available to be set based on the processing specifications of your e-forms.

Usage

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

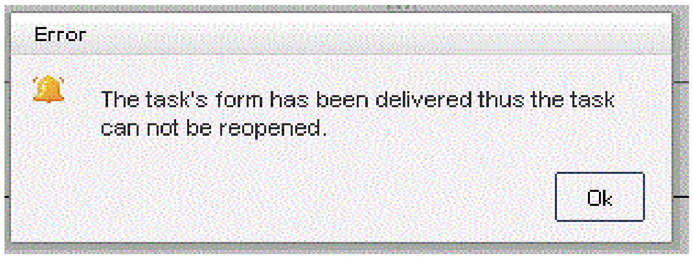
Note: After the form delivery status has been set to completed, the task associated with the form can not be reopened.

Parameters

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strFormID	Required. 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. 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.	simple string

After processing a form, if you choose to set the delivery status to “1”, the task with the associated e-form can not be reopened. This functionality is to insure the data is not updated after an e-form is delivered to the external application.

If the task is attempted to be reopened the follwoing message is displayed.



Error

The task's form has been delivered thus the task can not be reopened.

After processing a form, if you choose to set the delivery status to “2”, the task with the associated e-form can be reopened and the form can be updated. If the system accepting the forms data is intelligent to process 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 will be set back to 0 (zero). if you want the updated information to be updated to your external system, you should use this setting. This functionality is to insure form data can be updated after it is delivered.

Based on the requirements of the dependent application for e-form data changing after it has been delivered, you may choose to 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 it’s rules regarding one active E-Verify case per employee. RedCarpet has multiple checks to prevent an employer from opening mulitple 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 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 form 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 RedCarpet 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-opend 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.

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.

““ 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

Code examples

To set the delivery status of an e-form to “delivered” and check the error status:

```
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;

    }
```

Retrieve Populated PDF Files

RedCarpet forms can optionally be designed to merge the form data to an existing PDF file format.

Forms that have been configured with the corresponding PDF field mappings can be requested as a binary return format instead of a text (XML formatted) string.

GetFormPDF

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 both the form and the task (the form is associated with). The programmatic user will require ‘View All Tasks’ and ‘Manage Forms’ permission

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

Parameters

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strFormID	Required. A single formID	simple string

Returns

A table of type EpsBinaryEx.

The structure is:

```
EpsBinaryEx{
    String ErrorString;
    int ErrorNum;
    Byte[] 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.

““ 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)

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]);

FileStream f = File.OpenWrite(strFileName);

f.Write(arrBytes,0,arrBytes.Length);

f.Close();
```

Delete Specified Form

A delete method is provided for responsible management of your forms based on your specifications.

DeleteForm

Usage

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

Parameters

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strFormID	Required. A single formID	simple string

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 of less than 1, for 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

Code example

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);

  return;

  }
```

GetUploadedDocumentList

Usage

GetUploadedDocumentList returns a list of document id's. Documents returned include supporting documents uploaded for I-9's, 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 the specified search criteria.

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

Input for GetUploadedDocumentList

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strXML	Required XML is described below. The XML specifies the document criteria for selection.	simple string

Nodes in GetUploadedDocumentList strXML

Node	Description	Notes
`<?xml version="1.0" encoding="utf-8"?>`	XML Document header	Required
`<GetUploadedDocumentList Input>`	Parent node	

Node	Description	Notes

`<UploadDateRange>`
`<BeginDate>`
`</BeginDate>`
`<EndDate>`
`</EndDate>`
`</UploadDateRange>`
	Optional - date range of upload
	
	Occurs once.
		Begin Date and / or End Date can be specified.
		
		Fomat is YYYY-MM-DD

<DocumentOwnerFilter> <EmployeeIDFilter> <EmployeeIDType> </EmployeeIDType> <ID> </ID> </EmployeeIDFilter> </DocumentOwnerFilter>	Optional EmployeeIDType and ID occurs for as many Employee ID as to be returnd, (see example below)	Possible values for EmployeeIDType are: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteI

Node	Description	Notes
<KeyPropertyFilter> <CategoryValue> <CategoryCode/> <CategoryValueCode /> <Person> <PersonCode> <EmployeeID> <EmployeeIDType> </EmployeeIDType> <ID> </ID> </PersonCode> </Person> <Date> <DateCode> <DateRange> <BeginDate/> <EndDate/> </DateRange> </DateCode>	Optional <KeyPropertyFilter> Occurs once per request. Child nodes occur multiple time as applicable to the employee definition.	Possible values for EmployeeIDType are: Employee_HRISID LoginID SSOAuthParam Email Guid LifeSuiteI Date format is YYYY-MM-DD DateCode and PersonCode are specified as they are defined in Key Properties.
<DocumentTypeFilter> <DocumentType> </DocumentType> </DocumentTypeFilter/>	Optional. Document TypeFilter occurs once per call	DocumentType values are:: DocumentUpload EmployeeDocumentUpload I9Upload

Node	Description	Notes
</ 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 Document.

The structure is:

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

ErrorString

Return code description. The value of *ErrorNum* should be checked for a the value of less than 1, for 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

Sample Output

The following sample output returns two uploaded documents:

```
<?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"?>

<Documents xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

   <Document>

      <PageId>5416bafd-615c-43dc-bfd000bbbc77a2ed</PageId>

      <Type>jpg</Type>

      <DocumentType>I9Upload</DocumentType>

      <Owner>

         <Guid>240f972f-3349-406f-94866b20d1ea8f25</Guid>

         <Employee_HRISID>3004</Employee_HRISID>

         <SSOAuthParam />

         <LoginID>Dan.connolly</LoginID>

         <Email>dconnolly@mysrt.com</Email>

         <LifeSuiteID />

      </Owner>

   </Document>

   <Document>

      <PageId>abe67439-26d6-4246-82cb0331a546883a</PageId>

      <Type>pdf</Type>

      <DocumentType>I9Upload</DocumentType>

      <Owner>

         <Guid>56e6f1f7-aab1-43ca-9c5900c2b95d7087</Guid>

         <Employee_HRISID>1001</Employee_HRISID>

         <SSOAuthParam />

         <LoginID>Tammy.Jones</LoginID>

         <Email>tjones@MySRT.com</Email>

         <LifeSuiteID />

      </Owner>

   </Document>
</Documents>

]]></Data>

</EpsStringEx>
```

GetUploadedDocument

Usage

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

Input for GetUploadedDocumentList

strSecurityToken	Required. Valid Session ID for consuming service	simple string
strPageID	Required page Id returned by GetUploadedDocumentList .	simple string

Returns

An XML Document.

The structure is:

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

<EpsBinaryEx xmlns="http://Eprise">

   <ErrorString>No Error</ErrorString>

   <ErrorNum>1</ErrorNum>

   <Data><![CDATA ]></Data>

</EpsBinaryEx>
```

ErrorString

Return code description. The value of *ErrorNum* should be checked for a the value of less than 1, for 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.

Chapter 5: Category Upload

Overview

The Category Upload API allows a import and edit of the category values.

The XML syntax for creating and modifying category values is documented in a separate document called the Category Upload Guide (named “RCCategoryImport.pdf”). The Category Upload Guide describes the syntax and available operations to add and modify the category hierarchy. It also includes a description of two support features (Export and Restore Points) available to support category hierarchy changes.

Category Upload

An XML record used to create and edit category values.

CategoryUpload

Usage

Used to programmatically create or modify the category value hierarchy using an XML formatted file.

This web method does not run an initial validation pass on the XML.

Parameters

strSecurityToken	Required. Valid Session ID for consuming service	simple string
categoryXml	Required if uploadFileName is blank. XML record formatted as described in Chapter 2 of the Category Upload Guide.	string
uploadFileName	Optional. Required if categoryXml is blank	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.

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.

““ 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 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

Bulk Import Error Codes

The following table contains the possible error codes:

<Message>Value	<ErrorCode> Value
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

<Message>Value	**<ErrorCode> Value**
'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

Chapter 6: Reports API

Overview

Two API's are available to execute the RedCarpet Events report. GetEventReportEx (released with RedCarpet 2.9.0) and GetEventReport (depreciated with the release of GetEventReportEx).

GetEventReportEx

Usage

Used to generate XML output using the same parameters available in the RedCarpet user interface.

The columns included in the report output are specified in the RedCarpet user interface Settings page.

Parameters for GetEventReportEx2

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

Example Structure of GetEventReportEx2 strXML

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

<GetEventReportEx2Input>

<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>

</DateRange>

</GetEventReportExInput></GetEventReportEx2Input>
```

Nodes in GetEventReportEx2

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

Node	Description	Notes
`<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 ull 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>`
`<DateRange [type='Effective' (type attribute optional)]>`	Optional ((minOccurs = 0, maxOccurs = 1)	Valid types: Effective(default), Activation, Create, Due, Completed, HireDate) Example: `<DateRange type='HireDate'>`

Node	Description	Notes

`<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>`	close node	Required
`</GetEventReportExInput>`	close node	Required
`</GetEventReportEx2Input>`	close parent node	Required

Returns

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[<Events><Event><EffectiveDate>12/1/2012 12:00:00
AM</EffectiveDate><EventCreationDate>12/4/2012 2:24:37 PM</
EventCreationDate><EventCompletionDate>12/5/2012 10:43:10 AM</
EventCompletionDate><EventStatus>Complete</
EventStatus><EmployeeEmployee_ID /
><EmployeeEmail>msmith@MySRT.com</
EmployeeEmail><EmployeeHireDate>12/1/2012 12:00:00 AM</
EmployeeHireDate><EmployeeTerminationDate /
><EmployeeStatus>Active</EmployeeStatus><EmployeeFirstName>Mark</
EmployeeFirstName><EmployeeLastName>Smith</EmployeeLastName></
Event></Events>]]></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 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, RedCarpet will replace the special character with an _ (underbar). If the replacement results in a non-unique category name a number will be 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, GetUserProfileEx2 and GetTask methods for additional information.

GetEventReport

Usage

Used to generate XML output using the same parameters available in the RedCarpet user interface.

The columns included in the report output are specified in the RedCarpet user interface Settings page.

Parameters

Please note the strStartingDate and strEndingDate parameters have a default value to limit the data being returned.

strSecurityToken	Required. Valid Session ID for consuming service includes the Reports privilege	simple string
strEventName	Optional.	simple string
arrCategoryFilter	Optional Used to specify the benefiter's category values associated with the event. " " - (empty string) - all categories will be returned . valid values match the strings as specified on the Manage Hierarchical Category Values under Administration -> Manage Categories page in the RedCarpet user interface. See "Syntax of the Category Parameter" on page 81.	string ArrayOfArrayOfS tring

strStartingDate	Optional Optional. Used to specify the start of a date range. The date range queries the effective date of the event. If srtStartingDate is not specified it will defaul to yesterday. A string in yyyy-mm-dd format.	string
StrEndingDate	Optional Optional. Used to specify the end of a date range. The date range queries the effective date of the event. If srtEndingDate is not specified it will defaul to today A string in yyyy-mm-dd format.	string

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.

““ 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 import new RedCarpet users from an XML record.

```
Eprise.EpsStringEx ret = service.GetEventReport(sessionnum,”Onboarding”,,,);

if (ret.ErrorNum != 1)

{

    Console.Out.WriteLine("Error while execute Event Report" + " Error:" +
    ret.ErrorString);

    return;

    }
```

The ret.Data string contains the xml ouptut as is available in the RedCarpet user interface packaged into a CDATA string.

Sample return (with the ret.Data string) is a follows:

```
<?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"?><Events><Event><EmployeeName>Nathan Townsend</
EmployeeName><LoginID>Nathan.Townsend</
LoginID><EventName>Onboarding</EventName><EventID>18</
EventID><EffectiveDate><![CDATA[1/23/2009 12:00:00
AM]]]]><![CDATA[></EffectiveDate><EventCreationDate><![CDATA[1/20/
2009 11:51:41 AM]]]]><![CDATA[></
EventCreationDate><EventCompletionDate></
EventCompletionDate><EventStatus>Active</
EventStatus><EventCoordinatorName>RC2 HRCoord</
EventCoordinatorName><EventCoordinatorType>HR Coordinator</
EventCoordinatorType><TotalTasks>11</TotalTasks><CompleteTasks>4</
CompleteTasks><IncompleteTasks>6</IncompleteTasks><OverdueTasks>7</
OverdueTasks><InactiveTasks>1</InactiveTasks><WarningTasks>0</
WarningTasks><TaskTitle>Send Welcome Letter</
TaskTitle><TaskDefStringID>SWL</TaskDefStringID><TaskID>131</
TaskID><TaskDefinitionID>1</
TaskDefinitionID><CompletionDate><![CDATA[1/20/2009 11:52:13
AM]]]]><![CDATA[></
CompletionDate><ModeledActivationDate><![CDATA[1/20/2009 12:00:00
AM]]]]><![CDATA[></
ModeledActivationDate><ActivationTimestamp><![CDATA[1/20/2009
12:00:00 AM]]]]><![CDATA[></
ActivationTimestamp><DueDate><![CDATA[1/20/2009 12:00:00
AM]]]]><![CDATA[></DueDate><Assignee>RC2 HRCoord</
Assignee><AssigneeTeam>HR Coordinators</
AssigneeTeam><CompletedBy>RC2 HRCoord</CompletedBy></Event> ...

</Events>]]></Data>

</EpsStringEx>
```

Chapter 7: Using Transformations

Overview

RedCarpet methods allow you to use XSLTs to transform both the input and the output of many RedCarpet 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 RedCarpet method. RedCarpet allows a transformation of the output to another format for ease of use for consumption of the output to another system.

SilkRoad also uses the output XLSTs to maintain backward compatibility with previous versions of the RedCarpet methods.

Manage Transformations

Usage

The Manage Transformations page is available through the RedCarpet user interface. The manage page allows you to upload XSLT files into the RedCarpet database. Uploaded files are available to be referenced by the method call either by operation or by reference.

RedCarpet administrators with the “Manage Transformation” privilege have access the “Manage Transformations” Administration menu option.

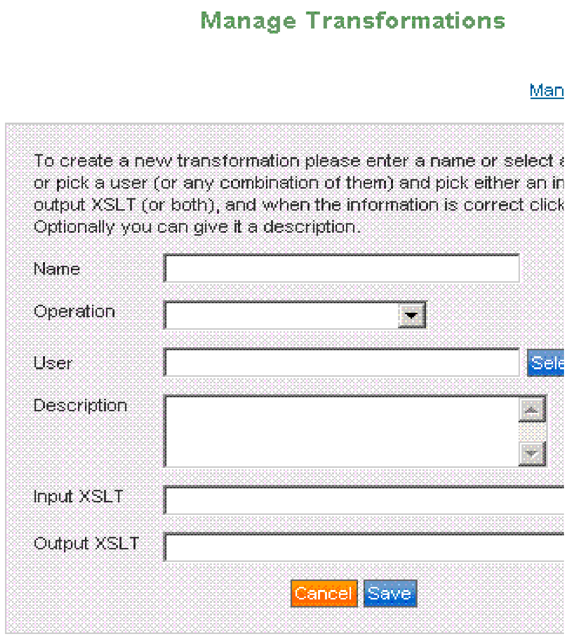
Defined transformations are applied based on reference by name, each time an operation (method) is executed, or based on which user is executing the method.

Default Configuration

RedCarpet 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 RedCarpet will work after upgrading to versions of RedCarpet after 2.7.x.

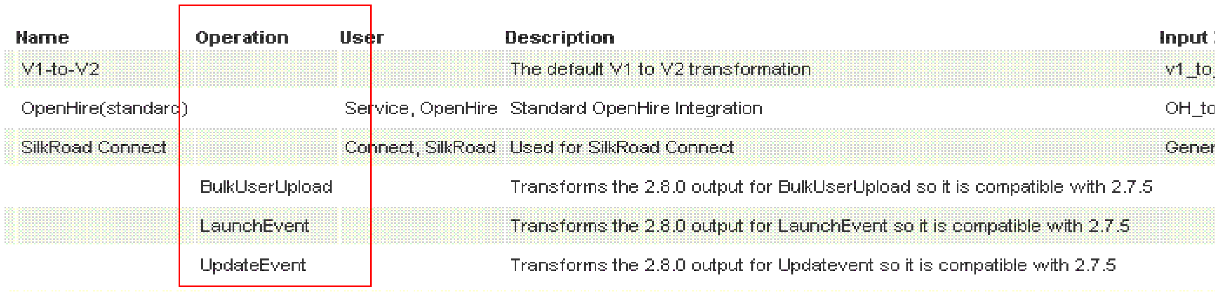
For customers to take advantage of updates to the output you must modify the transformation definition to remove the transformation of 2.8.0 output to 2.7.x output.


Manage Transformations

Man

To create a new transformation please enter a name or select a
or pick a user (or any combination of them) and pick either an in
output XSLT (or both), and when the information is correct click
Optionally you can give it a description.

Name	
Operation	
User	
Description	
Input XSLT	
Output XSLT	



Name	Operation	User	Description	Input
V1-to-V2			The default V1 to V2 transformation	v1_to_
OpenHire(standard)		Service, OpenHire	Standard OpenHire Integration	OH_to
SilkRoad Connect		Connect, SilkRoad	Used for SilkRoad Connect	Gener
	BulkUserUpload		Transforms the 2.8.0 output for BulkUserUpload so it is compatible with 2.7.5	
	LaunchEvent		Transforms the 2.8.0 output for LaunchEvent so it is compatible with 2.7.5	
	UpdateEvent		Transforms the 2.8.0 output for Updatevent so it is compatible with 2.7.5	

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 RedCarpet 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
- GetUserProfileEX2
- GetCompletedFormsList
- GetUploadedDocumentList
- ReopenTask
- GetFormXMLEx
- FieldValueUpload

Rules for Applying Transformations

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, RedCarpet shall determine 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,
    - RedCarpet shall search for the transformations matching that name, and any of Operation (the web method being handled) or Logged in user
    - RedCarpet shall use 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,
    - RedCarpet shall search for the transformations matching any of Operation (the web method being handled) or Logged in user.

- RedCarpet shall use 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 RedCarpet matches Operation above, RedCarpet shall first match Operation and Form Type, and then only Operation.

See RedCarpet Online Help for additional information on the Manage Transformation page.

Appendix A

Sample Code

Listed below is a C++ code example of the available method calls.

Each form download method is executed (tested) based on the booleans set on page 14.

```
using System;
using System.IO;
using System.Text;
using DownloadRCForms.Eprise;
namespace DownloadRCForms
{
/// <summary>
/// Summary description for Class1.
/// </summary>
class theApp
{
public static int FindObjectidColumn(String[]
columns)
{
  for (int i = 0 ; i < columns.Length ; i++){
  if (columns[i].IndexOf("ObjectId")==0)
  return i;
  }
return -1;
}
/// <summary>
/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main(string[] args)

{

try

{

 PSWebService service = new PSWebService();

 service.Url = "https://example.silkroadtech.com/eprise/
 WebServices";

 String sessionnum =
 service.LogIn("Jim.Forms_Admin","$123xyz$","");

 if (sessionnum == "")

 {

   Console.WriteLine("Could not log in.  Be sure password and URL
   are correct");

   return;

   }

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

int nObjectid = FindObjectidColumn(t.Data[0]);

if (nObjectid<0)

return;


                Directory.CreateDirectory("c:\\downloadedforms");

bool bDownloadXML = false;

bool bDownloadPDF = true;

bool bDeleteForm = false;

bool bMarkDelivered = false;

UTF8Encoding ue = new UTF8Encoding();


if (t.ErrorNum != 1){

 Console.Out.WriteLine("Error while trying to retrieve list of
 forms."  + t.ErrorString);

 return;

 }
```

```
for (int i = 1 ; i < t.Data.Length ; i++)

{

if (bDownloadXML)

{

   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][nObjec
  tid]);

  FileStream f = File.OpenWrite(strFileName);

  Byte [] arrBytes = ue.GetBytes(strXML);

                         f.Write(arrBytes,0,arrBytes.Length);

  f.Close();

  }

if(bDownloadPDF){

   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][nObjec
   tid]);

   FileStream f = File.OpenWrite(strFileName);

                         f.Write(arrBytes,0,arrBytes.Length);

   f.Close();
   }

if (bMarkDelivered)

  {

  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;

  }

}

if (bDeleteForm)

   {

   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);

   return;

   }

   }

}

}catch(Exception e){

  Console.Out.WriteLine(e.ToString());

  }

}

}

}
```