Cherwell REST API (CSM 2022.3)

Legal Notices

The Cherwell Software product suite includes:

- Cherwell Service Management
- Cherwell Asset Management

Contact Cherwell Software

Contents

Cherwell REST APIs

Cherwell REST APIs provide programmatic access to many CSM functions via an HTTP-based RESTful API. Operations are available for:

- Finding, creating, and updating Business Objects
- Finding and running Search queries
- Managing Users
- Getting Mobile Forms

About Cherwell REST APIs

The REST API Discovery Tool provides comprehensive API documentation with an intuitive user interface that enables you to discover and test operations using your CSM data. Testing is available for all authentication modes, except SAML.

REST API Revision History

CSM Version	Changes
10.2.2	Modified the **Add/Edit Canonical Definitions** dialog box to be read-only to allow canonical definitions to be maintained and provided only by Cherwell.
10.2.0	• Added operations for Business Object Lifecycle, including an operation for transitioning a Business Object record. • Added v2 of the storedsearches operation to provide greater flexibility for third-party reporting integrations. • Added optional "locale" and "lang" parameters to the Swagger UI, allowing the culture to be supplied for API requests from the Swagger UI. • Minor bug fixes.
10.1.0	Minor bug fixes.
10.0.0	• Added the ability to get the status of an Approval Business Object. • Added the ability to return an Approval Business Object using the Approval record ID. • Added the ability to get all waiting Approvals for the current user. • Added the ability to get all pending Approvals that were created by the current user.
9.7.0	• Added the ability to export all rows of data for Get Results of a Saved Search. • Introduced the ability to execute a One-Step™ Action or get a list of available One-Step Actions.
9.6.0	• Added the ability to connect to third-party tools to retrieve data from CSM using the Cherwell REST API. Configure the REST API by creating a new API Client ID (or use an existing one) to automatically generate a Client Key. The Client Key is used to generate an access token for the REST API. • Added authentication to the Service Monitor using the Cherwell REST API. The following configuration keys were added into the web.config for the REST API: UseSAMLADFSRedirect and IDPisADFS.

CSM Version	Changes
9.5.0	• Added the Canonical REST API Mapping Wizard to simplify the process of mapping a canonical Business Object Schema to your custom Business Object Schema. • Added operation for retrieving mapped Activity types for Audit, Communication, and Notes. • Removed installed documentation. Users are now directed to the version-specific REST API documentation on https://help.cherwell.com.
9.4.0	• Added Named Object Operations to perform basic functions against Business Objects via the Cherwell REST API. Named Object Operations include nine API operations with simplified URL paths to provide user-friendly, programmatic access to basic CSM functions. You may want to use the Named Object Operations if you are new to the Cherwell REST API or you do not have extensive programming experience. Functions that you can perform using the Named Object Operation include: ◦ Delete Business Object ◦ Get Business Object ◦ Update Business Object ◦ Create Business Object ◦ Get Business Object Field List ◦ Delete Business Object Comment ◦ Get Business Object Comments ◦ Create Business Object Comment ◦ Search for Business Object • Added the Cherwell Canonical REST API. The Cherwell Canonical REST API enables customers and partners to create reusable integrations between third-party software and customized CSM systems by providing a fixed schema to map out-of-the-box Business Objects to customized Business Objects.
9.3.0	Minor bug fixes.

CSM Version	Changes
9.2.0	• Added operations for: ◦ Removing Customers from Workgroups. ◦ Saving Workgroup Members and Team Members. ◦ Getting Teams using the Team ID. ◦ Deleting Teams; removing Users from Teams. ◦ Getting Users using the Record ID; getting a list of all Users. ◦ Adding items to Queues; checking items in to Queues. ◦ Removing items from Queues; checking items out of Queues. ◦ Getting Queues using Scope, Scope Owner, and Folder. ◦ Getting Stored Values using Scope, Scope Owner, and Folder. ◦ Creating, updating, and deleting gallery images. ◦ Getting gallery images using Scope, Scope Owner, and Folder. ◦ Setting the culture for the current User. • Added multiple Powershell examples to the documentation. Expanded information about using OAuth2 and SAML protocols with the REST API. Added SAML client examples with and without a final URI query string. Added practice exercises for using the REST API Discovery Tool. Reorganized the Operations List to match Swagger by adding new categories for Team Operations, User Operations, and Queue Operations. Added instructions for re-generating Swagger code with the new categories after upgrading.
9.1.0	• Added operations for: ◦ Creating and updating Teams and Workgroups. ◦ Adding a batch of Users to Teams. ◦ Getting a Business Object record based on its associated scan code. • Added support for HTML content in Business Object fields.
9.0.1	Added multiple C# examples to the documentation.
8.3.1	• Added v2 of most Security operations to change Error property to ErrorMessage. • Added v3 of getuserbyloginid to change Error property to ErrorMessage.

CSM Version	Changes
8.2.1	• Added operations for: ◦ Executing Quick Searches. ◦ Running Saved Queries that have Prompts. ◦ Adding Users to Teams; removing users from Teams. ◦ Getting Mobile Forms. • Improved Landing Page that includes parameter glossary and Swagger code-gen instructions.
8.1.1	• Added v2 of getuserbyloginid. • Changed API version to match CSM version.
8.1.0	• Support for Windows, LDAP, and SAML authentication. • Attachment management and built-in image retrieval operations added.
8.00	Initial Release.

About the Cherwell Canonical REST API

The Cherwell Canonical REST API allows for reusable integrations between third-party software and customized CSM systems by providing a fixed schema to map out-of-the-box Business Objects to customized Business Objects.

For example, for a system with a custom Business Object called Ticket, you can leverage an integration with a the OOTB Incident Business Object by using the Canonical REST API to map from the Incident Schema to the Ticket Schema.

The Canonical REST API is primarily intended for creating reusable integrations between third-party software and customized CSM systems. If you need to integrate with your instance of CSM, we recommend beginning with the Named Object Operations.

Important: To ensure standardization, canonical definitions are supplied and managed by Cherwell.

View Canonical Definitions

View canonical definitions provided by Cherwell.

1. In CSM Administrator, create a Blueprint.
2. Select **Managers > Canonical Definitions**.
3. Select a folder to see canonical definitions in your system.
4. Use the Canonical Mapping Wizard to map fields in definitions to fields in your custom Business Object. See Canonical REST API Mapping Wizard.

Canonical REST API Mapping Wizard

Prepare your system for third-party integrations with a custom Business Object by mapping a canonical Business Object schema to your custom Business Object schema using the Canonical REST API Mapping Wizard. You can then use the Canonical REST API to programmatically access CSM functions for your custom Business Object.

You can also use the Mapping Wizard to overwrite existing mappings at any time. If you use the Auto Mapping feature, the Wizard attempts to overwrite all existing mappings for that canonical definition.

Important: To ensure standardization, canonical definitions are supplied and managed by Cherwell.

To use the Canonical Mapping Wizard:

1. In CSM Administrator, navigate to **Tools > Canonical Mapping Wizard**
   The **Map Canonical Object Wizard** opens.
2. Select **Next**, and then select an imported definition from the **Canonical Definition** list.
3. From the **Business Object** list, select the Business Object you want to map to.
4. Use one of these options to create the mappings:
   a. **Auto Map**: Select this button for the system to attempt to map Canonical fields to Business Object fields based on the field name. The field names are case-sensitive and must exactly match for the auto mapping to work.
   If the Wizard does not find an exact field name match, the drop-down menu for that field will show that it is not mapped. You will need to manually select a field.
   b. **Manually Map**: Choose a mapping for each field from the drop-down menu.

   Note: You must map each field in the canonical definition.
5. On the Summary page, select the **Return extended errors** check box to return detailed information in HTTP requests that use the canonical definition.
6. Select **Finish** to create a Blueprint for the mapping.
7. Repeat these steps for each canonical definition you want to map.
8. Once you have mapped the fields for each canonical definition, publish the Blueprint, then run a Health Check for canonical compliance to ensure your mapping was successful.

Run a Health Check for Canonical Compliance

The Health Check for canonical compliance evaluates mappings between canonical Business Object schemas and custom Business Object schemas. It returns errors if the mappings are incomplete or incorrect. Use the Health Check to verify that your system is ready to connect to the Canonical REST API.

1. From the CSM Administrator main window, select the **Performance** category.
2. Select **Run Health Check**.
3. Clear all boxes except **Check Canonical Compliance**.
4. Select **OK**.
   The Health Check Results open in a separate window. Select **Save to File** to save the results to an HTML file. Select **Submit** to send the results to Cherwell Support on request.

Interpreting the Results of a Canonical Compliance Health Check

Use the results of your canonical compliance Health Check to verify whether you successfully mapped a canonical Business Object Schema to a custom Business Object Schema. The Health Check results indicate fields that were mapped successfully as well as fields that encountered errors.

When you run the Health Check for canonical compliance, your results open in a new window. The results show each mapping that is defined between a canonical Business Object Schema and a custom Business Object schema.

Results in black indicate successful mappings and show the mapped fields for each schema. Results in red indicate errors with one or more field mappings.

Common errors include:

Error Message	Description
Missing one or more field maps: [Field]	The canonical Business Object Schema includes the indicated field, and it is not mapped to a field in the custom Business Object Schema.
[Field] => [Field] (does not exist)	The mapped field does not exist. Verify that the field exists and ensure the correct spelling and case.
[Field](Type) => [Field](Type) (FieldType Mismatch)	The mapped fields are not of the same type; for example, you cannot map a logical field to a number field.
[Field](Source field required but not mapped)	The custom Business Object Schema includes the indicated, required field, and it is not mapped to a field in the canonical Business Object Schema.
[Field](RecId) not mapped to canonical	The custom Business Object Schema includes a Record ID field that is not mapped to a field in the canonical Business Object Schema.

Canonical REST API Usage and Examples

The Canonical REST API supports basic CRUD (create, read, update, and delete) operations on Business Objects and Business Object comments. You can also perform search operations to search for Business Objects. Code samples for the Canonical REST API are available in C#.

The Canonical REST API is primarily intended for creating reusable integrations between third-party software and customized CSM systems. If you need to integrate with your instance of CSM, we recommend beginning with the Named Object Operations.

Before you begin programming, verify that your system is set up to connect to the Canonical REST API. See Canonical REST API Mapping Wizard.

C# code samples for the Canonical REST API are provided on the Cherwell GitHub page in the CanonicalAPI-Samples repository.

Opening the REST API Discovery Tool

Become familiar with the REST API Discovery Tool (Swagger) before you begin coding so that you can build a program that interacts effectively with CSM.

The Cherwell REST API Discovery Tool (Swagger) can be found by appending the following parameters to the CSM site name URL:

/CherwellApi/swagger/ui/index

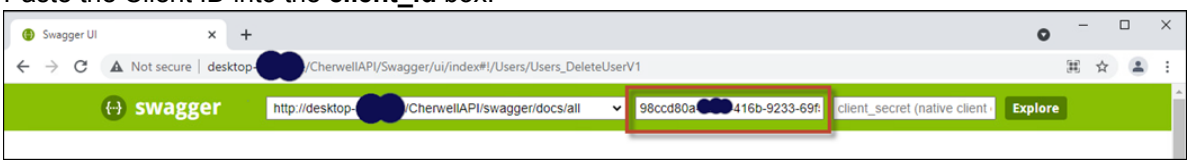
Logging in to the Discovery Tool

To log in to the Discovery Tool, you need:

- An API client ID. See Obtaining API Client IDs.
- A CSM login ID and password.
- An API Client Secret ID (reserved for Cherwell Software use only).

To log in to the Discovery Tool:

1. Go to the URL for YourCSMSiteName/CherwellApi/swagger/ui/index in a browser.
2. Paste the Client ID into the **client_id** box.



3. Select the **Authentication** button for a specific operation, and then provide your CSM credentials.

Tip: An error of "Is not registered" means that the 36 character Client ID is invalid. See Obtaining API Client IDs for instructions on generating a new one.

Practice Using the REST API Discovery Tool

Become familiar with the REST API Discovery Tool (Swagger) before you begin coding so that you can build a program that interacts effectively with CSM. Practice calling operations and interpreting the responses so that you are prepared to write and troubleshoot your program.

No experience with APIs is necessary to perform the practice exercises, though some programming experience will make it easier for you to grasp the concepts. A strong background in CSM is highly recommended. Work closely with your system administrator if your CSM experience is limited.

Important: Use a test environment to perform practice exercises.

Getting the Most From the Practice Exercises

To maximize your understanding, we recommend that you progress through the exercises in order of difficulty, beginning with the simplest:

1. Practice Exercise: Get Search Items
2. Practice Exercise: Get Business Object Schema
3. Practice Exercise: Get Quick Search Configuration for Business Objects
4. Practice Exercise: Save User

Practicing on Your Own

In addition to the provided practice exercises, you may practice calling any of the API operations on your own by inputting the required parameters and generating JSON responses. To help interpret response data, consider using an online JSON viewer.

Practice Exercise: Get Search Items

This is the first in a series of practice exercises to help you become familiar with Cherwell's REST API Discovery Tool. In this exercise, you will call a simple operation with a single, boolean parameter.

1. Log into the REST API Discovery Tool. See Opening the REST API Discovery Tool.
2. Click the **Searches** heading to expand the list of operations.
3. Click **/api/V1/getsearchitems** to expand the operation's details.
4. Under parameters, select **true** for the links parameter.
5. Click **Try it out!**
   If the response data does not appear, ensure that the authentication button has been switched to **On**.
6. **Optional:** Try calling the operation again using **false** for the links parameter to see the difference in response data.

Practice Exercise: Get Business Object Schema

This is the second in a series of practice exercises to help you become familiar with Cherwell's REST API Discovery Tool. We highly recommend that you complete the following exercises before you begin:

- Practice Exercise: Get Search Items

In this exercise, you will call the Get Business Object Summary operation to get the Business Object ID for Incident. Then, you will use the Business Object ID to call the Get Business Object Schema operation.

Get the Business Object ID:

1. Log into the REST API Discovery Tool. See Opening the REST API Discovery Tool.
2. Click the **Business Objects** heading to expand the list of operations.
3. Click **/api/V1/getbusinessobjectsummary/busobname/{busobname}** to expand the operation's details.
4. Under parameters, type Incident for the busObName parameter.
5. Click **Try it out!**
   If the response data does not appear, ensure that the authentication button has been switched to **On**.
6. From the response body, copy the value for busObId. This is the Business Object ID that you will use to call the next operation.

Use the Business Object ID to get the Business Object schema:

7. Click **/api/V1/getbusinessobjectschema/busobid/{busobid}** to expand the operation's details.
8. Under parameters, paste the Business Object ID in the busObId Value field.
9. Select **false** for the includeRelationships parameter so that the response data will not include information about related Business Objects.
10. Click **Try it out!**
11. **Optional:** Try calling the operation again using a different Business Object ID or including information about related Business Objects.

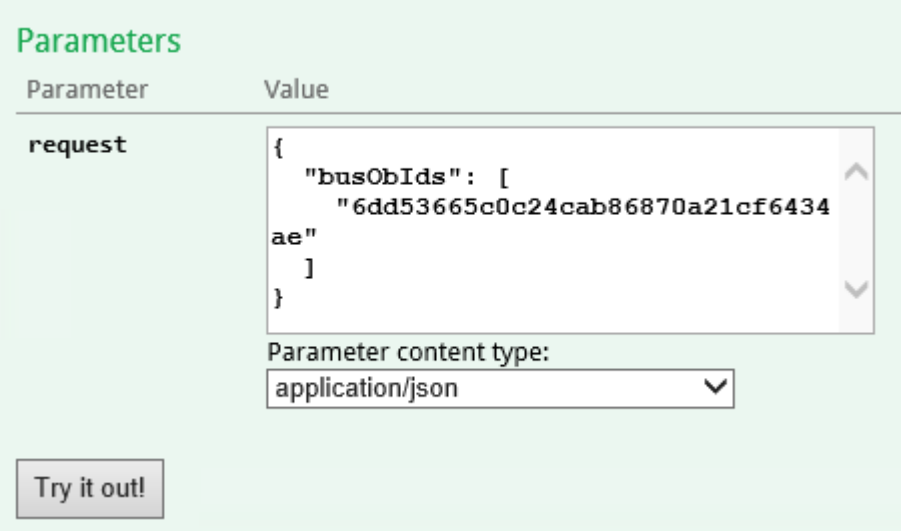
Practice Exercise: Get Quick Search Configuration for Business Objects

This is the third in a series of practice exercises to help you become familiar with Cherwell's REST API Discovery Tool. We highly recommend that you complete the following exercises before you begin:

- Practice Exercise: Get Search Items
- Practice Exercise: Get Business Object Schema

In this exercise, you will use a Model Schema to build a basic request parameter, which will be passed into the API call.

1. Log into the REST API Discovery Tool. See Opening the REST API Discovery Tool.
2. Click the **Searches** heading to expand the list of operations.
3. Click **/api/V1/getquicksearchconfigurationforbusobs** to expand the operation's details.
4. Under Data Type, ensure **Model Schema** is selected. Click the box under Model Schema to set its contents as the parameter value.
   The request Value field should match the Model Schema.
5. In the request Value field, replace the word string with the Business Object ID for Incident.



   For the procedure to get a Business Object ID, see Practice Exercise: Get Business Object Schema.

6. Click **Try it out!**
   If the response data does not appear, ensure that the authentication button has been switched to **On**.
7. **Optional:** Try calling the operation again using a different Business Object ID.

Practice Exercise: Save User

This is the fourth in a series of practice exercises to help you become familiar with Cherwell's REST API Discovery Tool. We highly recommend that you complete the following exercises before you begin:

- Practice Exercise: Get Search Items
- Practice Exercise: Get Business Object Schema
- Practice Exercise: Get Quick Search Configuration for Business Objects

In this exercise, you will use a Model Schema to build a complex request parameter, which will be passed into the API call in order to create a new User.

1. Log into the REST API Discovery Tool. See Opening the REST API Discovery Tool.
2. Click the **Users** heading to expand the list of operations.
3. Click **/api/V2/saveuser** to expand the operation's details.
4. Set the contents of the request Value field to match the model schema.

In the next steps you will edit the JSON code as you did for Practice Exercise: Get Quick Search Configuration for Business Objects. You can edit the code directly in Swagger, or you can copy the code into a text editor so that it is easier to work with.

5. Set the value for accountLocked to **false**.
6. Get the Business Object ID for UserInfo.
   For the procedure to get a Business Object ID, see Practice Exercise: Get Business Object Schema.
7. Set the value for busObId to the Business Object ID for UserInfo
8. Delete the key-value pairs for busObPublicId and busObRecId from the JSON code. These values are not required for the creation of a new User.
9. Set the value for displayName to **Test User**.
10. Set the value for ldapRequired to **false**.
11. Set the value for loginId to **Test**.
12. Set the value for nextPasswordResetDate to **null**.
13. Set the value for password to **P@ssword**. Ensure this value meets the password complexity requirements set in CSM.
14. Ensure the value for passwordNeverExpires is set to **true**.
15. Set the value for securityGroupId to the Security Group ID for the User's Security Group. For this exercise, choose any Security Group.
    Call **api/V2/getsecuritygroups** to see a list of Security Groups and their IDs.
16. Ensure the value for userCannotChangePassword is set to **true**.
17. Set the values for userInfoFields. Call **/api/V1/getbusinessobjectschema/busobid/{busobid}** to get the schema for the UserInfo Business Object. For each field:
    a. Ensure the value for dirty is set to **true**.
       This setting must be enabled for each field in order to save the field values.

b. Set the value for displayName to the Display Name.
c. Set the value for fieldId to the Field ID.
d. Set the value for name to the Name.
e. Set the value for the field.

18. Set the value for userMustChangePasswordAtNextLogin to **false**.
19. Delete the key-value pair for windowsUserId. You are creating a Cherwell User, so this value is unnecessary. See the example below of the full request body:

```
{
  "accountLocked": false,
  "busObId": "9338216b3c549b75607cf54667a4e67d1f644d9fed",
  "displayName": "Test User",
  "ldapRequired": false,
  "loginId": "Test",
  "nextPasswordResetDate": "null",
  "password": "P@ssword",
  "passwordNeverExpires": true,
  "securityGroupId": "93d5aa70c8893014a2b7dc4733953c6185777e92bf",
  "userCannotChangePassword": true,
  "userInfoFields": [
    {
      "dirty": true,
      "displayName": "Full Name",
      "fieldId": "BO:9338216b3c549b75607cf54667a4e67d1f644d9fed,FI:9338
2178280a07634f62d74fc4bc587e3b3f479776",
      "name": "FullName",
      "value": "Test User"
    },

    {
      "dirty": true,
      "displayName": "Email",
      "fieldId": "BO:9338216b3c549b75607cf54667a4e67d1f644d9fed,FI:9338
21793f43a638cf23e34723b907956d324ad303",
      "name": "Email",
```

```
      "value": "user@domain.com"
    }
  ],
  "userMustChangePasswordAtNextLogin": false,
}
```

20. Click **Try it out!**
    Pay attention to the HTTP response codes you receive, and troubleshoot as appropriate:

200	Indicates a successful call.
400	Indicates a bad request due to missing parameters or incorrect formatting.
402	Indicates a failed call. The request body is formatted correctly, but the system is unable to execute the request.
404	Indicates a failed call due to a nonexistent URL.

21. **Optional:** Try adding or updating field values for the User you just created.

Operations List

Cherwell REST API operations are organized by category and ordered by URL.

Most of the operations in this list are presented using Swagger UI. The documentation is generated from a demo build of CSM. The operations that are documented in addition to the Swagger UI topic are user-configured and outside the scope of our default environment.

This section is intended to provide general information about the Cherwell REST API. You can explore operations that are specific to your instance of CSM using the Rest API Discovery Tool.

Canonical Operations

Refer to About the Cherwell Canonical REST API for information about the Cherwell Canonical REST API.

These operations are primarily intended for customers and vendors to create reusable integrations between third-party software and customized CSM systems. If you are a Cherwell Administrator and need to integrate with your instance of CSM, we recommend beginning with the Named Object Operations.

Note: For the operation URLs listed below, `{BusObName}` is replaced in Swagger with the canonical name of the Business Object on which the action is performed, and `{Version}` is replaced with the version of the canonical Business Object Schema. For example, for a custom Business Object called Ticket that is mapped to version one of the canonical Schema for Incident, the Describe Business Object URL is `GET /api/V1/canonical/Incident`.

Operations are ordered by URL.

Describe Business Object

- **URL**
  `GET /api/{Version}/canonical/{BusObName}`
- **Description:**
  Get the Business Object Schema using the canonical Business Object name.

Create Business Object

- **URL**
  `POST /api/{Version}/canonical/{BusObName}`
- **Description:**
  Create a new Business Object.

Delete Business Object by Business Object ID

- **URL**
  `DELETE /api/{Version}/canonical/{BusObName}/{ID}`
- **Description:**
  Delete a Business Object using the Business Object ID.

Update Business Object by Business Object ID

- **URL**
  `PATCH /api/{Version}/canonical/{BusObName}/{ID}`

- **Description:**
  Update a Business Object using the Business Object ID.

Search for Business Object by Search Term, Value, or Date Range

- **URL**
  `GET /api/{Version}/canonical/{BusObName}/search`
- **Description:**
  Search for a Business Object using a search term, value, or date range. You can specify one or more search criteria. The search parameters are appended to the URL.

Search for Business Object

- **URL**
  `POST /api/{Version}/canonical/{BusObName}/search`
- **Description:**
  Search for a Business Object. You can specify one or more search criteria. The search parameters are contained within the request body.

Get Business Object Comments

- **URL**
  `GET /api/{Version}/canonical/{BusObName}/{id}/Comments`
- **Description:**
  Get comments associated with a Business Object using the Business Object ID.

Create a Business Object Comment

- **URL**
  `POST /api/{Version}/canonical/{BusObName}/{id}/Comments`
- **Description:**
  Create a comment for a Business Object using the Business Object ID.

Delete a Business Object Comment

- **URL**
  `POST /api/{Version}/canonical/{BusObName}/Comments/{id}`
- **Description:**
  Delete a comment for a Business Object using the comment ID.

Describe Business Object Comments

- **URL**
  `GET /api/{Version}/canonical/{BusObName}/Comments`
- **Description:**
  Get the Schema for the Business Object comments.

Named Object Operations

Use Named Object Operations to perform basic actions against Business Objects. Named Object Operations include nine API operations with simplified URL paths that you can run against any major Business Object.

Named Object Operations are not automatically visible in Swagger. To add the Named Object Operations for a specific Business Object, follow the steps to Add Operations for Named Objects.

Important: For the operation URLs listed below, `{BusObName}` is replaced in Swagger with the internal name of the Business Object on which the action is performed. For example, for Incident, the Delete Business Object URL is `DELETE /api/V1/object/Incident/{id}`.

Operations are ordered by URL.

Delete Business Object

- **URL**
  `DELETE /api/V1/object/{BusObName}/{id}`
- **Description:**
  Delete a Business Object using its name and ID.

Get Business Object

- **URL**
  `GET /api/V1/object/{BusObName}/{id}`
- **Description:**
  Get a Business Object using its name and ID.

Update Business Object

- **URL**
  `PATCH /api/V1/object/{BusObName}/{id}`
- **Description:**
  Update a Business Object using its name, ID, and properties.

Create Business Object

- **URL**
  `POST /api/V1/object/{BusObName}`
- **Description:**
  Create a Business Object using its name and properties.

Get Business Object Field List

- **URL**
  `GET /api/V1/object/{BusObName}/describe`
- **Description:**
  Get a Business Object Field list using its name.

Delete Business Object Comment

- **URL**
  `DELETE /api/V1/object/{BusObName}/{id}/comments/{commentid}`
- **Description:**
  Delete a Business Object comment using its name, ID, and comment ID.

Get Business Object Comments

- **URL**
  `GET /api/V1/object/{BusObName}/{id}/comments`
- **Description:**
  Get a list of comments for a Business Object using its name and ID.

Create Business Object Comment

- **URL**
  `POST /api/V1/object/{BusObName}/{id}/comments`
- **Description:**
  Create a comment for a Business Object using its name and ID.

Search for Business Object

- **URL**
  `POST /api/V1/object/{BusObName}/search`
- **Description:**
  Search for a Business Object using its name and an array of filters.

More Information

See the Introducing the Named Object REST API free Video Learning Library course.

Add Operations for Named Objects

Add the PublishToApi attribute to a Business Object so that its REST API operations become available in Named Object Operations.

1. Open CSM Administrator, and create a new Blueprint.
2. Select a Business Object from the Object Manager.
3. Click **Edit Business Object** under Business Object tasks.
   The **Edit Business Object** page appears.
4. Click the **Bus Ob Properties** button.
   The **Business Object Properties** dialog opens.
5. Click **Advanced** and expand **General Attributes**.
6. In the **Attribute** column, type **PublishToApi**. Leave the corresponding **Value** field empty.
7. Click **OK** and save the Blueprint.
8. Repeat steps 2 through 7 for additional Business Objects, if necessary.
9. Publish the Blueprint.
   REST API operations for the selected Business Objects are available in the Named Object Operations section of Swagger.

Securing the REST API

CSM supports the OAuth2 protocol to authenticate and authorize calls to the REST API. The OAuth2 protocol enables third-party clients to obtain access to HTTP services. In basic OAuth2 message flow, the client interacts with a resource server and an authorization server. The client requests tokens from the authorization server on behalf of the user. The authorization server authenticates the User and returns tokens to the client. The client then sends the tokens to the resource server, which hosts the protected resource, and the User is able to access the resource as long as the tokens remain valid. In our environment, CSM acts as both the resource server and the authorization server.

You may optionally configure your system to support the SAML authentication protocol for the REST API. When SAML authentication is configured, CSM no longer acts as the authorization server in the OAuth2 message flow. Instead, your SAML identity provider acts as the authorization server. The identity provider receives token requests from the client, authenticates Users, and returns tokens to authorize access to the Cherwell REST API.

Obtaining API Client IDs

Client IDs are created in CSM Administrator.

Note: You can create separate client IDs to control access for specific users and specific integration tools.

1. In CSM Administrator, select **Security**.
2. Select **Edit REST API client settings**.
3. Select the **plus** (+) icon.
4. Provide these settings to generate a client ID:

Setting	Description
Name	Provide a name for the client ID.
Culture	Select a language-specific culture.
Description	Provide a description for the client ID.
Token lifespan	Set the amount of time the access token will be active.
Refresh Token lifespan	Set the amount of time the refresh token will be active.
API access is enabled	When selected, the client ID is enabled. When cleared, the client ID is disabled. Clear the check box to disable the client ID without deleting it.
Allow anonymous access	Select to make the REST API available to anonymous users.

5. Select **Save**.
   The client ID is autogenerated for you and appears in the **Client Key** field. It is a 36 character string of randomly generated letters and numbers.
6. Copy the client ID and provide it to REST API users.

Set the Base URL for the Cherwell REST API

The base URL for the Cherwell REST API is required for various CSM features, including authentication, Saved Searches, and webhooks. It is automatically populated when you install CSM. If you need to change it, you can set the base URL for the REST API using Cherwell Server Manager.

You can also view, and in rare cases, change settings for webhooks used with Amazon Web Services (AWS) and Slack.

Setting the Base API URL

To set the base API URL:

1. In Cherwell Server Manager, select the **Configure** button by **Rest API client configuration**. The **REST API URL Settings** window appears.
2. Edit the URL.
3. In CSM Administrator main window, select **Security**.
4. Select **Edit REST API client settings**.
5. On the **REST API Clients** menu bar, select **File > Set API URL**
6. In the **Maximum Webhook Content Length** box, set the maximum number, in bytes, of body content that can be added by any webhook endpoint in the system. The default setting is 100,000 bytes; the maximum is 2,147,483,647 bytes.

   Note: If the content of any webhook POST exceeds the length, the webhook POST request will fail.

7. Select **OK**.

Viewing Amazon Webhook Settings

Webhooks settings used by Amazon Simple Notification Service (SNS) are provided. You should not modify these settings unless changes are required from Amazon Simple Notification Service (SNS).

If you modify settings and need to revert to the provided settings, select **Reset Webhooks Defaults**.

Setting	Description
Header Message Type	Determines the type of message being sent from AWS (notification; subscription; unsubscribe). Provided value: x-amz-sns-message-type
Subscription Confirmation	Sent in the header by AWS to indicate that the message type is a Subscription Confirmation.
Notification	Sent in the header by AWS to indicate that the message type is a notification.
Header Message ID	Sent by AWS to indicate message uniqueness to determine if messages have already been sent.

View Slack Webhook Settings

Webhooks settings used by Slack are provided. You should not modify these settings unless changes are required from Slack.

If you modify settings and need to revert to the provided settings, select **Reset Webhooks Defaults**.

Setting	Description
Signature Header Key	The name of a key sent in the header by Slack. The Slack signature computes the hash that verifies the message's authenticity.
Timestamp Header Key	The name of a key sent in the header by Slack. The timestamp is used to verify the message's authenticity.
Version	The version of the Slack API.
Replay Attack Timeout in Minutes	The number of minutes that elapse between a message being sent from Slack and being processed by CSM. This prevents an attacker from capturing a valid message, altering the contents, and then sending it later.

Related concepts

Cherwell REST API Command-Line Options

Related tasks

Configure Logging for the Cherwell REST API

OAuth2 Protocol for the REST API

All clients follow a basic message flow to access the Cherwell REST API using OAuth2. To begin, a user must obtain a client ID from CSM Administrator. Client IDs contribute to the security of the REST API by providing unique keys that work in conjunction with a User's CSM privileges.

After obtaining the client ID, the User performs a full login to the REST API using the client ID and their CSM User credentials. A successful login generates an access token, which allows the User agent to access the REST API as long as the token remains valid.

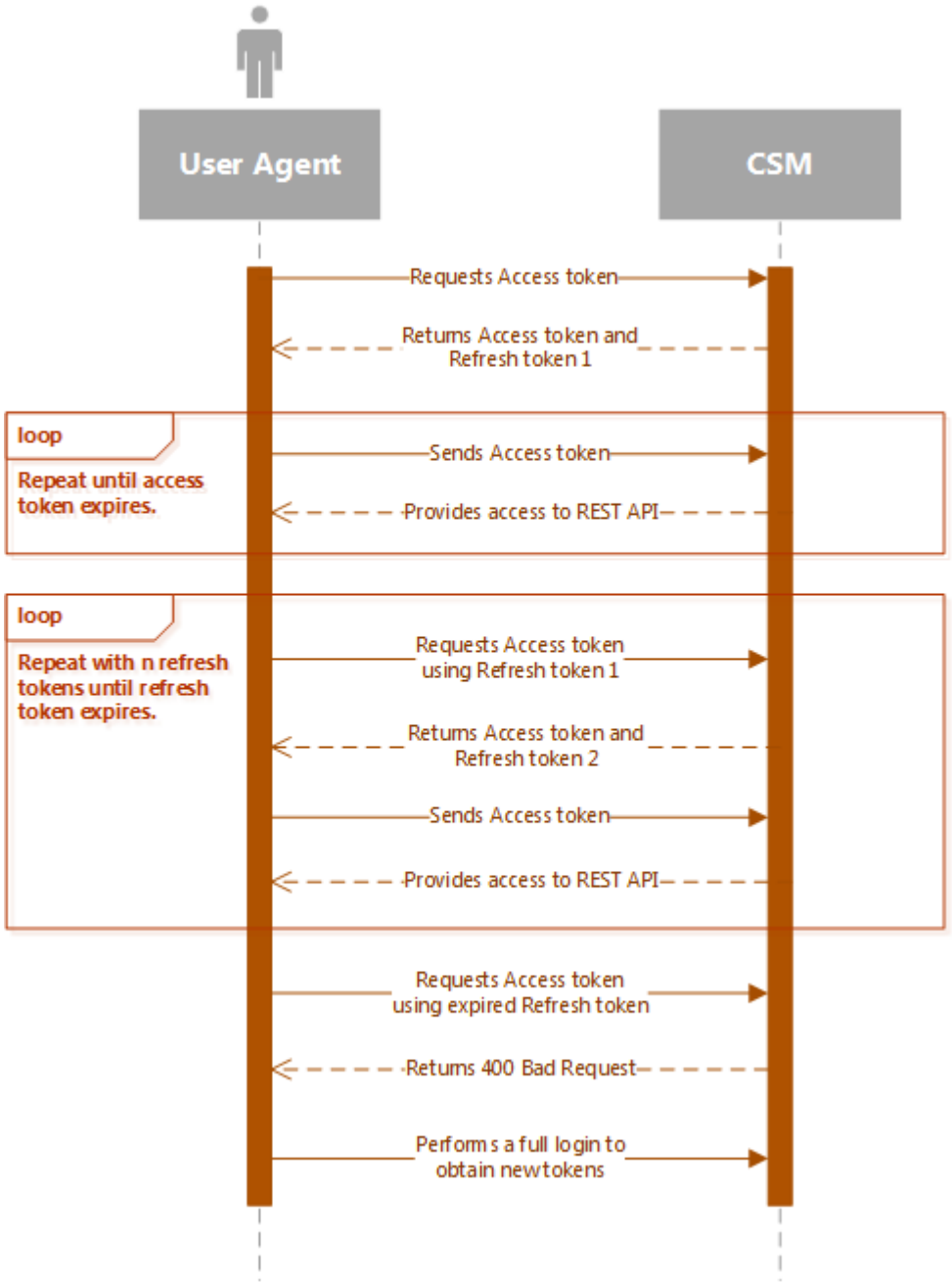
Note: The security settings that are configured for your Users will remain in effect when they access the REST API.

An Access token is needed for all subsequent calls to the REST API to identify the User agent as a valid API user. The life span of the Access token is based on the client ID's settings. Access tokens tend to have a relatively short life span and can be refreshed with a Refresh token.

Refresh tokens are used to periodically refresh the Access token without the need to provide credentials again. Refresh tokens tend to have a longer life span than Access tokens and are also based on the client ID's settings. Once the Refresh token has expired or the user is logged out, a full login must be performed to obtain new tokens. It is not always necessary to use Refresh tokens, which are generally considered safer over non-encrypted transport protocols, such as HTTP. If you use an HTTPS connection, you can increase the Access token life span and use Access tokens for all subsequent connections.

Note: To further secure your system, use SSL.

Users who log in to the REST API client consume a CSM license.


User Agent
CSM
Requests Access token
Returns Access token and Refresh token 1
loop
Repeat until access token expires.
Sends Access token
Provides access to REST API
loop
Repeat with n refresh tokens until refresh token expires.
Requests Access token using Refresh token 1
Returns Access token and Refresh token 2
Sends Access token
Provides access to REST API
Requests Access token using expired Refresh token
Returns 400 Bad Request
Perform s a full login to obtain new tokens

OAuth2 Authentication Modes

The REST API uses the authentication types specified for the CSM Browser Client:

- **Internal**

  Uses the login ID and password specified for a user in CSM. If no other mode is specified, Internal mode is used.

- **LDAP**

  Uses the LDAP settings configured for CSM and the server variable LOGON_USER to attempt to find a CSM user. You can also use domain\username and password.

- **SAML**

  Uses the SAML settings configured for CSM to validate credentials and find the CSM user.

- **Windows**

  Uses the server variable LOGON_USER to attempt to find a CSM user. You can also use domain\username and password.

Make an HTTP POST call to the *token* operation to get an access token. Pass data in to the request body according to a specific authentication type. CSM returns a JSON response that includes information about the access token. The example below shows the response body for the internal authentication mode:

```
{
  "access_token": "SampleAccessTokenValue",
  "token_type": "bearer",
  "expires_in": 1199,
  "refresh_token": "SampleRefreshTokenValue",
  "as:client_id": "SampleClientIdValue",
  "username": "Username",
  ".issued": "Fri, 31 Mar 2017 15:31:39 GMT",
  ".expires": "Fri, 31 Mar 2017 15:51:39 GMT"
}
```

Authenticating Using the Internal Mode

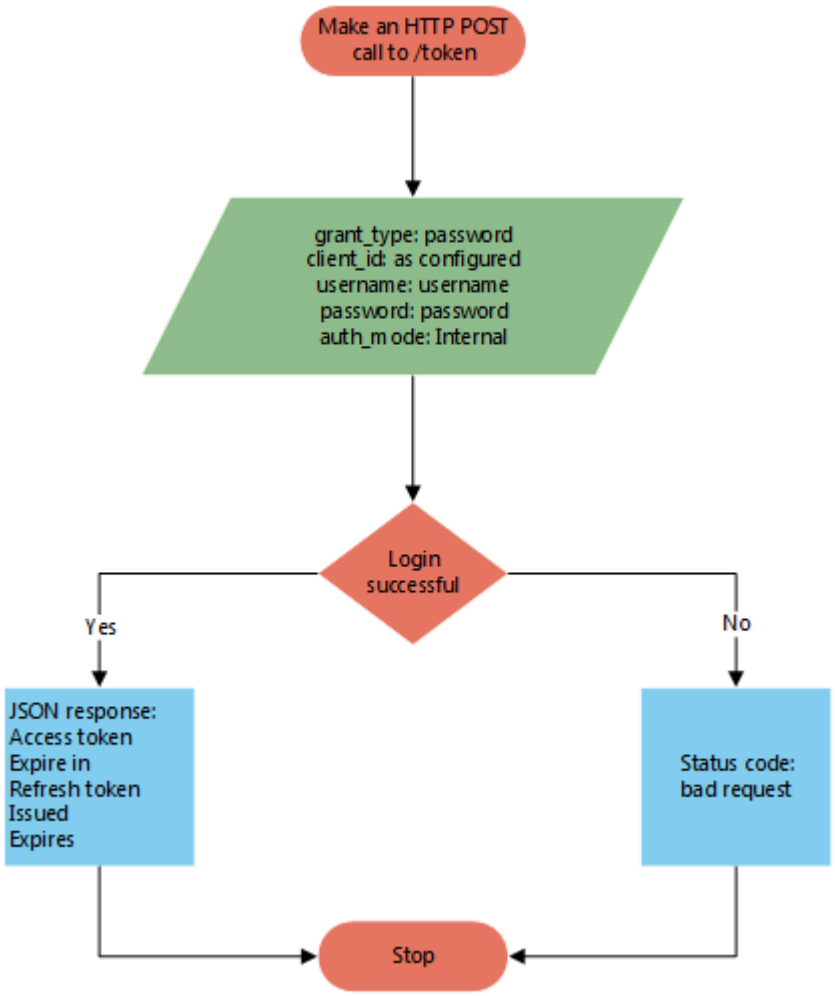
In this scenario, the User logs in to the REST API using CSM credentials.

1. Make an HTTP POST call to the token operation.
2. Pass data into the request body:

Parameter	Value
grant_type	password
client_id	as configured
username	username
password	password
auth_mode	internal

3. CSM attempts to log in the User using the supplied credentials.

   If the login is successful, an Access token and Refresh token are granted. If the login is unsuccessful, an error is returned.


Make an HTTP POST call to /token
grant_type: password
client_id: as configured
username: username
password: password
auth_mode: Internal
Login successful
Yes
No
JSON response:
Access token
Expire in
Refresh token
Issued
Expires
Status code: bad request
Stop

Authenticating Using the Client-Based Windows/ LDAP Mode

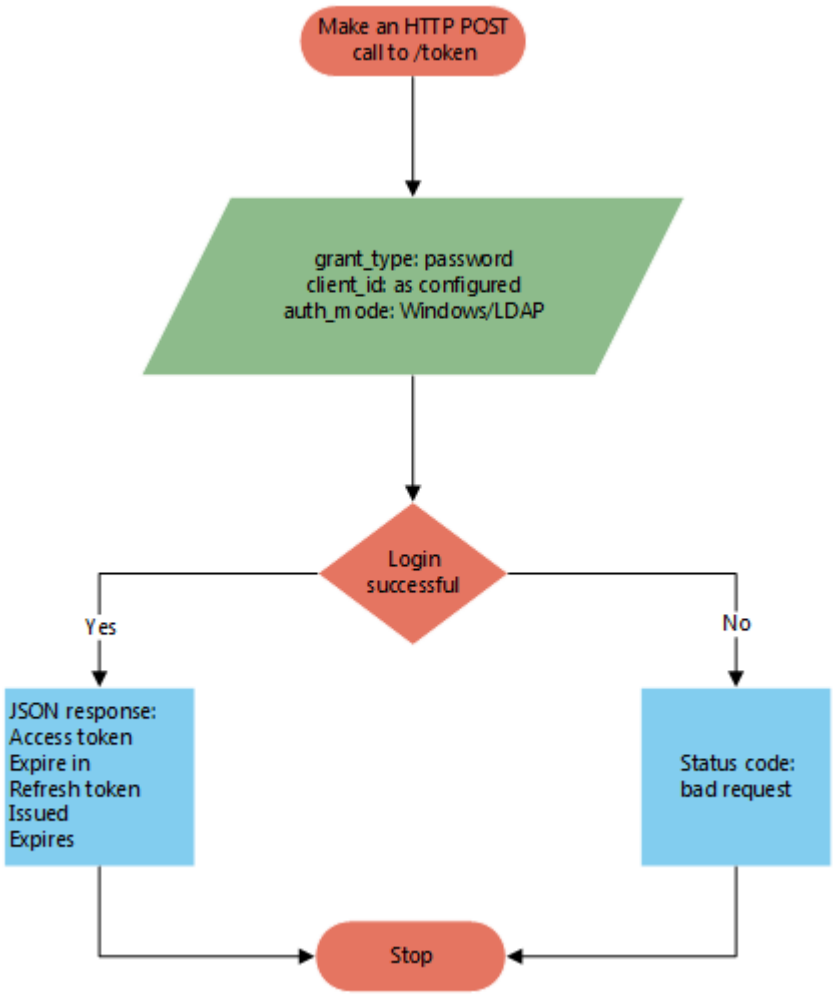
In this scenario, the User is already logged in to the Internet Information Services (IIS) manager using Windows credentials. CSM attempts to leverage the Windows ID and password to log in the User to the REST API so that entering additional credentials is not required. A successful login means the same User is logged in to both the operating system and the REST API.

1. Make an HTTP POST call to the token operation.
2. Pass data into the request body:

Parameter	Value
grant_type	password
client_id	as configured
auth_mode	Windows/LDAP

3. CSM attempts to log in the User using the Windows credentials leveraged from IIS.

   If the login is successful, an Access token and Refresh token are granted. If the login is unsuccessful, an error is returned.


Make an HTTP POST call to /token
grant_type: password
client_id: as configured
auth_mode: Windows/LDAP
Login successful
Yes
No
JSON response:
Access token
Expire in
Refresh token
Issued
Expires
Status code: bad request
Stop

Authenticating Using the Windows/LDAP Mode With Entered Credentials

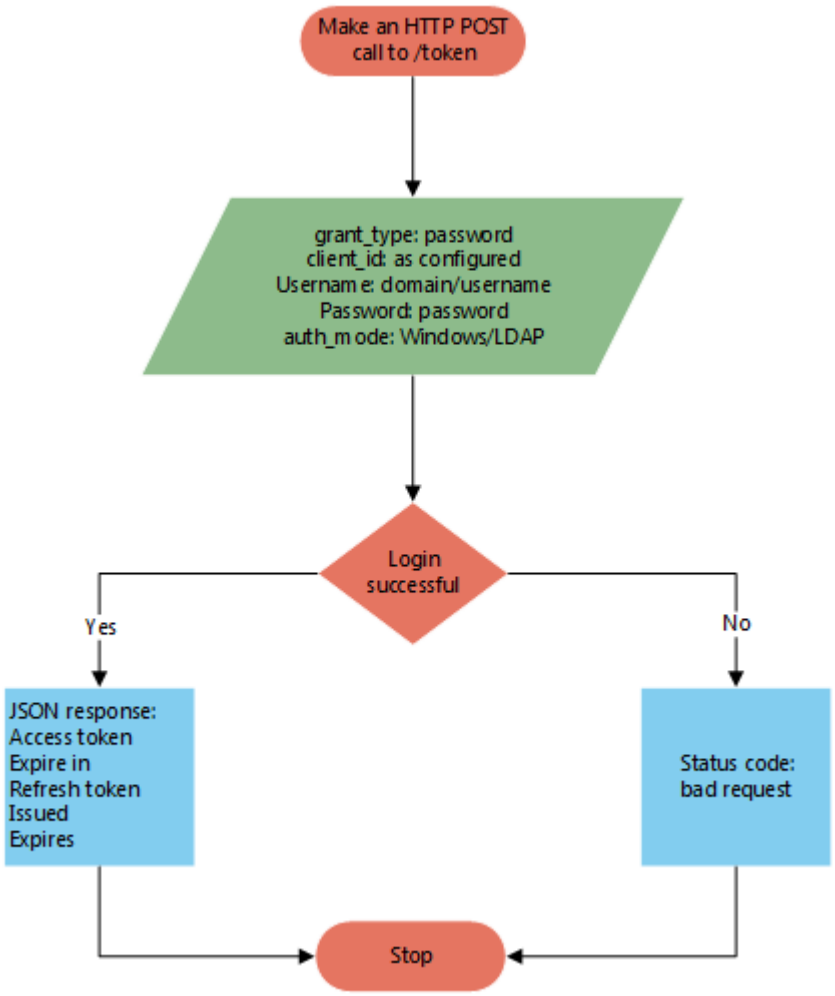
In this scenario, the User overrides CSM's attempt to log in using Windows user credentials from the IIS manager by passing in alternative Windows credentials. A successful login means a different user is logged in to the REST API than the operating system.

1. Make an HTTP POST call to the token operation.
2. Pass data into the request body:

Parameter	Value
grant_type	password
client_id	as configured
username	domain/username
password	password
auth_mode	Windows/LDAP

3. CSM attempts to log in the User using the supplied credentials.

   If the login is successful, an Access token and Refresh token are granted. If the login is unsuccessful, an error is returned.


Make an HTTP POST call to /token
grant_type: password
client_id: as configured
Username: domain/username
Password: password
auth_mode: Windows/LDAP
Login successful
Yes
No
JSON response:
Access token
Expire in
Refresh token
Issued
Expires
Status code: bad request
Stop

Requesting Access and Refresh Tokens

CSM returns a new Access and Refresh token and invalidates the Access and Refresh token that were previously in use.

Set the Token lifespan and the Refresh Token lifespan for each client ID when you obtain an API Client ID.

1. Make an HTTP POST call to the token operation to get an Access token. Pass data into the request body according to a specific authentication type.

   The example below shows the response body for the internal authentication mode:

```
{
  "access_token": "SampleAccessTokenValue",
  "token_type": "bearer",
  "expires_in": 1199,
  "refresh_token": "SampleRefreshTokenValue",
  "as:client_id": "SampleClientIdValue",
  "username": "Username",
  ".issued": "Mon, 17 June 2019 15:31:39 GMT",
  ".expires": "Mon, 17 June 2019 15:51:39 GMT"
}
```

2. Make an HTTP POST call to the token operation to get an Access token using a Refresh token.
3. Pass data into the request body:

Parameter	Value
grant_type	refresh_token
client_id	as configured
refresh_token	refresh token received with access token

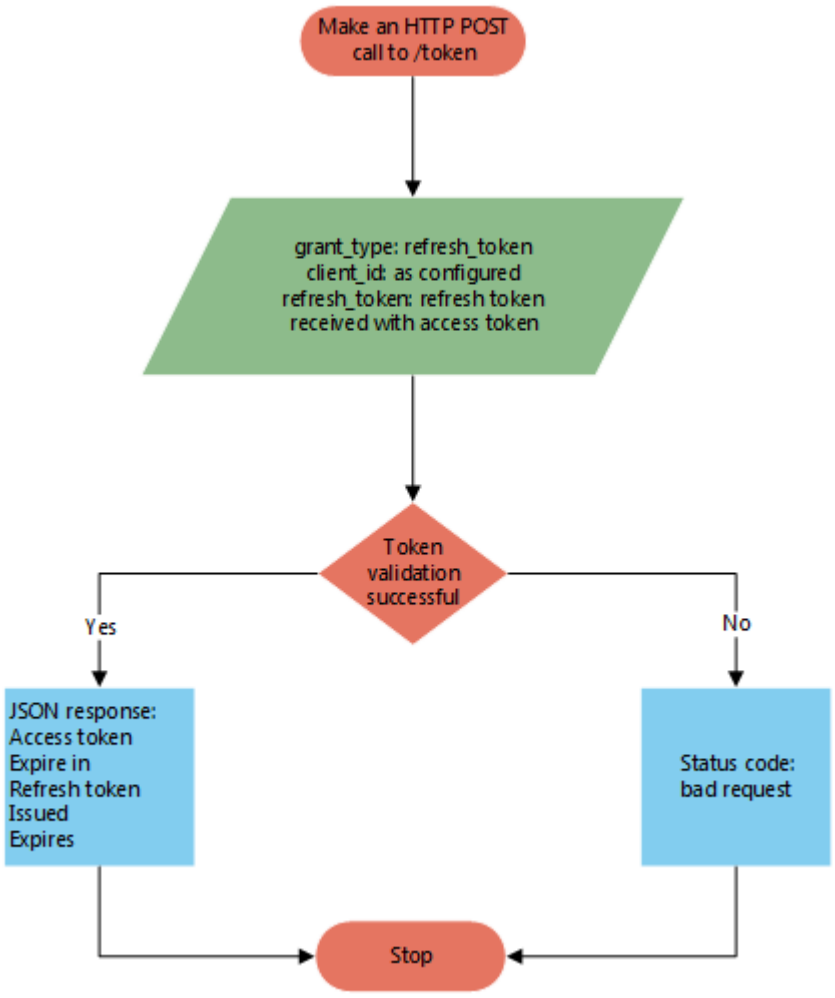
4. CSM validates the client ID and Refresh token using the token store.

   If the validation is successful, a new Access token and new Refresh token are returned.

```
{
  "access_token": "SampleAccessTokenValue",
  "token_type": "bearer",
  "expires_in": 1199,
```

```
  "refresh_token": "SampleRefreshTokenValue",
  "as:client_id": "SampleClientIdValue",
  "username": "Username",
  ".issued": "Mon, 17 June 2019 20:31:39 GMT",
  ".expires": "Mon, 17 June 2019 20:51:39 GMT"
}
```

If the validation is unsuccessful, an error is returned.



Related concepts

OAuth2 Authentication Modes

Related tasks

Obtaining API Client IDs

Get Access Token using Postman

You can use your preferred API testing tool, such as Postman, to obtain an access token for the Cherwell REST API.

To obtain an access token using Postman:

1. Create a new request in Postman.
2. Select the **POST** method.
3. Enter **http://yourservername/CherwellApi/token** for the server address, substituting your own server name or ip address.
4. In the body of the POST request, enter the following, substituting your own values for the parameters in the table.

```
grant_type=password&client_id={YourClientID}&username={YourUsername}&pa
ssword={YourPassword}
```

Parameter	Value
grant_type	"password"
client_id	As generated using CSM Administrator. See Obtaining API Client IDs.
username	Username.
password	Password for the username.

5. CSM validates the client ID using the token store.
   An access token is returned in the response section.

```
{
  "access_token": "SampleAccessTokenValue",
  "token_type": "bearer",
  "expires_in": 1199,
  "refresh_token": "SampleRefreshTokenValue",
  "as:client_id": "SampleClientIdValue",
  "username": "Username",
  ".issued": "Mon, 17 June 2019 15:31:39 GMT",
  ".expires": "Mon, 17 June 2019 15:51:39 GMT"
}
```

Related tasks
Requesting Access and Refresh Tokens

Obtaining API Client IDs

SAML Protocol for the REST API

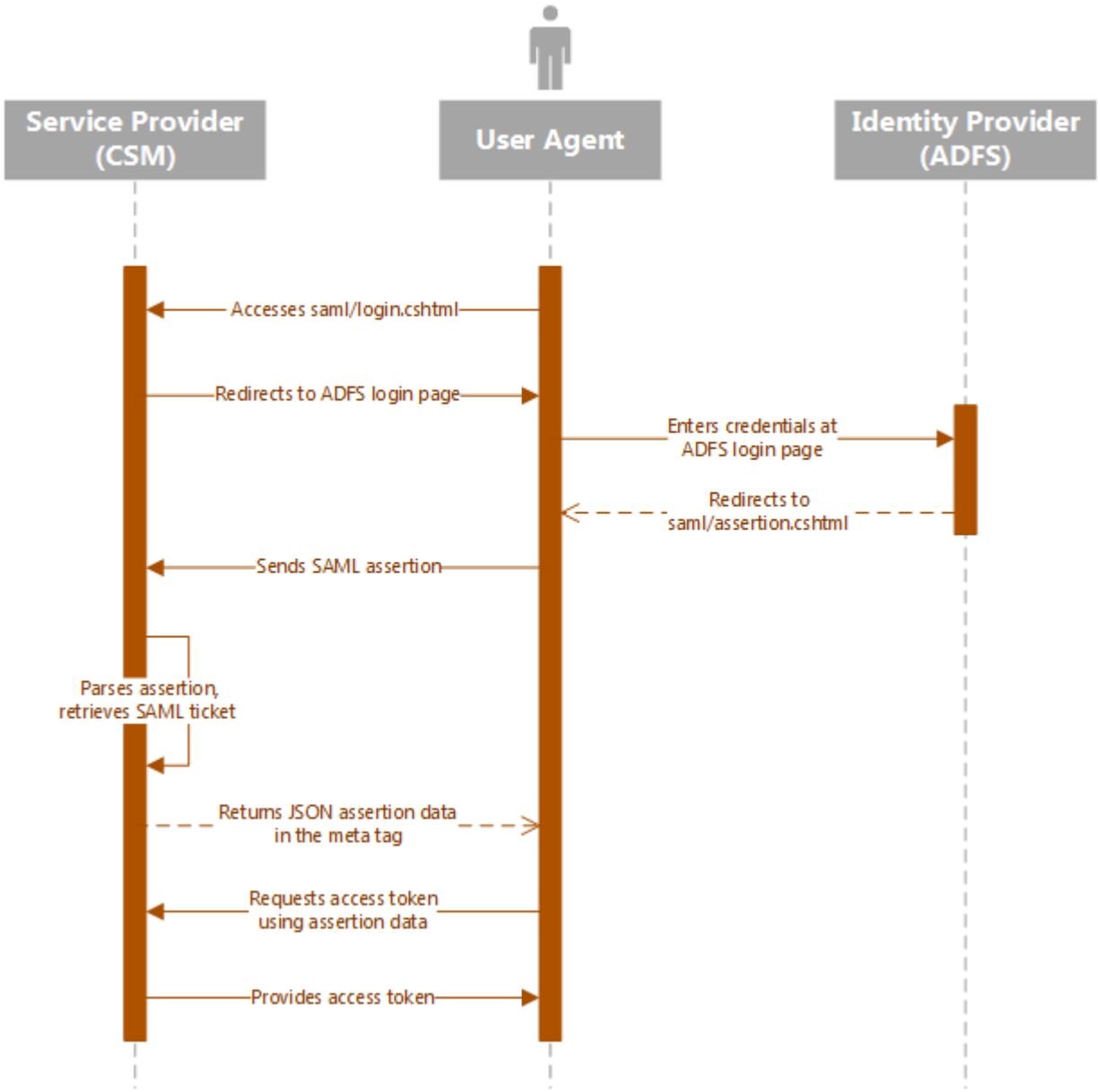
All clients follow a basic message flow to access the REST API using SAML. Whereas CSM acts as both the service provider and the identity provider in OAuth2 protocol, SAML protocol introduces a third-party identity provider.

For information about supported identity providers and the configuration procedure, see Configure the SAML Identity Provider

When a User navigates to the REST API, CSM redirects the User agent to the single sign-on service at the User's identity provider where the User enters his credentials. The User's identity is authenticated by the identity provider. If the authentication is successful, the identity provider returns a SAML assertion to CSM. The assertion indicates that a trusted identity provider successfully authenticated the User so that CSM can proceed to grant access to the REST API. CSM parses the assertion and returns the assertion data to the User agent. The User agent uses the assertion data to make a request to the token operation, and passes data in the request body as shown:

Parameter	Value
grant_type	password
client_id	as usual
username	e-mail address
password	parsed SAML ticket
auth_mode	SAML

If the request is successful, CSM returns an access token to the User agent. The access token allows the User agent to access the REST API as long as the token remains valid. The following diagram shows the basic SAML message flow:



SAML Protocol Using a Final URI Query String

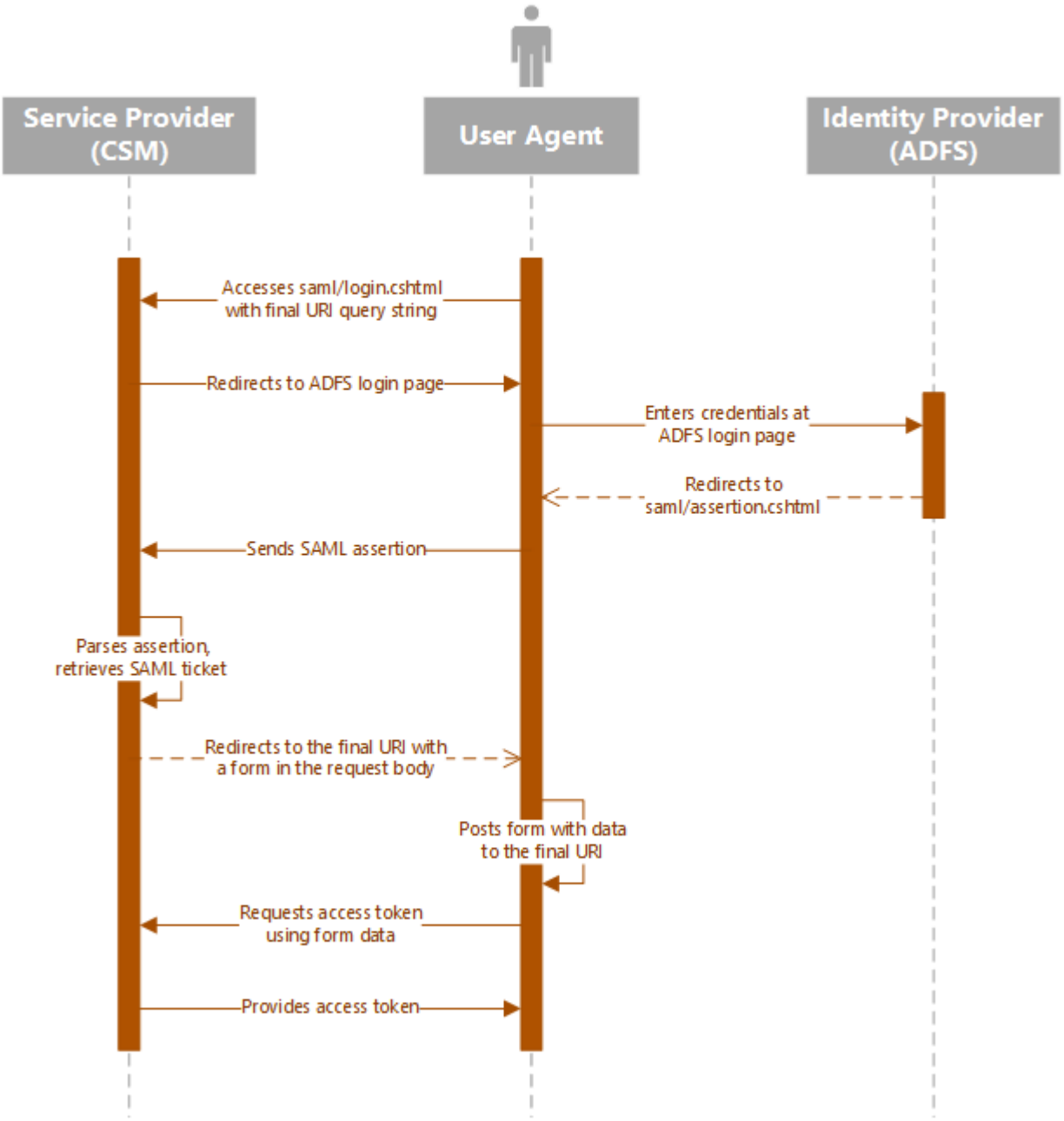
You may optionally use a final URI query string to pass form data to token requests. Example:

```
saml/login.cshtml?finalUri=http://localhost/SamlSampleBrowserApp/default.aspx?testQueryParameter=myValueForMySystem
```

After the SAML assertion is parsed, a web form is returned with an auto-submit method on the body of the form. The method posts form data to the requested final URI. Example:

```
<html>
<body onload='document.forms[""samlResult""].submit()'>
<form name='samlResult' action='{finalUri}' method='post'>
<input type='hidden' name='userId' value='{nameId}'>
<input type='hidden' name='nameQualifier' value='{nameQualifier}'>
<input type='hidden' name='ticket' value='{ticket}'>
<input type='hidden' name='result' value='ok'>
<input type='hidden' name='statusCode' value='{statusCode}'>
<input type='hidden' name='statusMessage' value='{statusMessage}'>
</form>
</body>
</html>
```

The following diagram shows the SAML message flow with a final URI:


Service Provider (CSM)
User Agent
Identity Provider (ADFS)
Accesses saml/login.cshtml with final URI query string
Redirects to ADFS login page
Enters credentials at ADFS login page
Redirects to saml/assertion.cshtml
Sends SAML assertion
Parses assertion, retrieves SAML ticket
Redirects to the final URI with a form in the request body
Posts form with data to the final URI
Requests access token using form data
Provides access token

Client Example: No Final URI

The following shows how to create a client for handling SAML protocol without a final URI:

```
webBrowser.Navigate("http://host/CherwellApi/saml/login.cshtml");

private void OnNavigated(object sender, NavigationEventArgs e)
{
 dynamic doc = webBrowser.Document;

 // Look for SAMLPOSTBACK meta tag - this is how we know we have the correc
t page
 var elements = doc.getElementsByTagName("meta");
 foreach (HTMLMetaElement element in elements)
 {
  if (string.Compare(element.name, "SAMLPOSTBACK", StringComparison.Ordinal
IgnoreCase) == 0)
  {
// if this were a popup dialog - could hide the dialog here and just retur
n the content to a calling page

   // the json data is stored in the metadata
   string samlAssertionData = element.content;
   LoginToRest(samlAssertionData);
  }
 }
}

private void LoginToRest(string samlAssertionData)
{
 var samlAssertion = JObject.Parse(samlAssertionData);
 // Setup the default header
 var baseuri = "http://host/CherwellApi/";
 var apiversion = "1";
 var apiClient = new ApiClient(baseuri);
```

```
 string clientid = "your client id";
 string user = samlAssertion.GetValue("NameId", StringComparison.InvariantC
ultureIgnoreCase).ToString();
 string password = samlAssertion.GetValue("Ticket", StringComparison.Invari
antCultureIgnoreCase).ToString();


 try
 {
  // Login to REST service
  ServiceApi service = new ServiceApi(apiClient);
  TokenResponse tokenResponse = service.ServiceToken("password", clientid,
user, password, string.Empty, "saml");
  // Setup the default header
  apiClient.DefaultHeader.Add("Authorization", "Bearer " + tokenResponse.Ac
cessToken);

  // Create search results request
  SearchResultsRequest searchResults = new SearchResultsRequest { BusObId
= "6dd53665c0c24cab86870a21cf6434ae" };

  // Query for data
  SearchesApi searches = new SearchesApi(apiClient);
  SearchResultsResponse response = searches.SearchesGetSearchResultsAdHocV
1(searchResults);
  string message = "There are " + response.TotalRows + " incidents";
  MessageBox.Show(message);
 }
 catch (ApiException ex)
 {
  MessageBox.Show(ex.Message);
 }
}
```

Client Example: With Final URI

The following shows how to create a client for handling SAML protocol with a final URI:

```
protected void Page_Load(object sender, EventArgs e)
{
 if (Request.Form.HasKeys() && Request.Form.Keys.Cast<string>().Any(x => x
== "ticket"))
 {
  var userId = Request.Form["userId"];
  var nameQualifier = Request.Form["nameQualifier"];
  var ticket = Request.Form["ticket"];
  var result = Request.Form["result"];
  var statusCode = Request.Form["statusCode"];
  var statusSubCode = Request.Form["statusSubCode"];
  var statusMessage = Request.Form["statusMessage"];

  var clientId = ConfigurationManager.AppSettings["clientId"];
  string baseuri = ConfigurationManager.AppSettings["apiBaseUri"];

  // Login to CherwellApi
  ServiceApi service = new ServiceApi(baseuri);
  TokenResponse tokenResponse = service.ServiceToken("password", clientId,
string.Empty,
              userId, ticket, string.Empty,
              "SAML");

  // Create search results request
  SearchResultsRequest searchResults = new SearchResultsRequest
  {
   BusObId = "6dd53665c0c24cab86870a21cf6434ae"
  };

  // create new instance of the Searches operations
  SearchesApi searches = new SearchesApi(baseuri);
```

```
  // Setup the default header for authorization
  searches.Configuration.DefaultHeader.Add("Authorization", "Bearer " + tok
enResponse.AccessToken);

  SearchResultsResponse response = searches.SearchesGetSearchResultsAdHocV
1(searchResults);

  // display results indicating success
  TotalRows.Text = "There are " + response.TotalRows + " incidents";

  var dict = Request.QueryString.Keys.Cast<string>().ToDictionary(key => ke
y, key => Request.QueryString[key]);
  var json = new JavaScriptSerializer().Serialize(dict);

  QueryParameters.Text = json;
 }
 else
 {
  // should be known if a user is attempting a SAML login to your system
  // make SAML login request to the CherwellApi
  Response.Redirect(ConfigurationManager.AppSettings["apiSamlLoginUri"]);
 }
}
```

Usage and Examples

The following topics provide usage and examples for specific operations.

Business Object Usage and Examples

The Cherwell REST API supports basic CRUD (create, read, update, and delete) operations on all Business Objects.

Usage: Field Identifiers

CSM field identifiers come in two forms:

- **Field identifiers for the main business object:**
  BO:6dd53665c0c24cab86870a21cf6434ae,FI:c1e86f31eb2c4c5f8e8615a5189e9b19
- **Field Identifiers for mapped relational fields:**
  FI:9397b3c0357e0678f9e3c94ce290e3783e4a61cabc,

  BO:9337c2311b8e8044aa3e2a48c4a95a9f5555815126,

  RE:9402756722d89a20ab008c41cc882fd219da37dc5f

Once a field has been created, its ID will not change.

Usage: Batch Operations

You can perform batch CRUD operations that allow you build up a collection of SaveRequests, ReadRequests or DeleteRequests on the client, and then send them as a batch to be processed on the server. This can greatly improve performance.

As batch operations are being processed, if one item in the batch fails to save, the entire batch stops processing if StopOnError is set to true. If StopOnError is set to false, the system will continue processing the remaining items in the batch.

When the system finishes processing the batch, a batch response is returned to the client. Inside of that object there is a collection of responses so you can determine if each batch operation failed or succeeded.

Usage: Creating and Updating Business Objects

Use these operations to create or update Business Objects:

- savebusinessobject
- savebusinessobjectbatch
- saverelatedbusinessobjects

To create Business Objects, do not include a record ID or public ID in the request.

To update existing Business Objects, specify the record ID or public ID in the request. If duplicate public IDs exist, you must use the record ID.

C# Example: Get an Incident

The following example shows how to get an Incident using C#.

Note: Run Swagger Code Generation before attempting to modify the code so that you do not receive reference errors.

```
using IO.Swagger.Api;

namespace Trebuchet.WebApi.IntegrationTests.ExamplesForCustomers.GetanObjec
t
{
    public class GetAnObject
    {
        public void GetABusinessObject()
        {
            //Get an access token using CSM credentials
            var serviceApi = new ServiceApi("https://your server/CherwellAp
i/");
            var tokenResponse = serviceApi.ServiceToken("password","your cl
ient id", null, "CSDAdmin", "CSDAdmin", null,"Internal");

            //Create a new Business Object api object and add the default h
eader
            var businessObjectApi = new BusinessObjectApi("https://your ser
ver/CherwellApi/");
            businessObjectApi.Configuration.AddDefaultHeader("Authorization
","Bearer " + tokenResponse.AccessToken);

            //Get the Business Object summary by name.  This returns the Bu
siness Object ID
            var businessObjectSummaryByName = businessObjectApi.BusinessObj
ectGetBusinessObjectSummaryByNameV1("Incident");

            //Get the Business Object by public ID
```

```
            var readResponse = businessObjectApi.BusinessObjectGetBusinessO
bjectByPublicIdV1(businessObjectSummaryByName[0].BusObId, "102667");
        }
    }
}
```

Related tasks

Using Swagger Code Generation

PowerShell Example: Get an Incident

The following example shows how to get an Incident using PowerShell.

```
# Set server login variables
$serverName = "serverName"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "https://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Get the business object summary for Incident. This will give us the busOb
Id
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/Inciden
t"
$summaryResults = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentTy
pe application/json -Header $requestHeader
$busObId = $summaryResults[0].busobId
```

```
# Get the business object by publicId
$getIncidentUri = $baseUri + "api/V1/getbusinessobject/busobid/${busobid}/p
ublicid/102522"
```

C# Example: Create an Incident

The following example shows how to create an Incident using C#.

Note: Run Swagger Code Generation before attempting to modify the code so that you do not receive reference errors.

```
using System.Collections.Generic;
using System.Linq;
using IO.Swagger.Api;
using IO.Swagger.Model;

namespace Trebuchet.WebApi.IntegrationTests.ExamplesForCustomers.Create_an_
Object
{
    public class CreateAnObject
    {
        public void CreateAnIncident()
        {
                                                                            /
/Get an access token using CSM credentials
            var serviceApi = new ServiceApi("http://your server/CherwellApi
/");
            var tokenResponse = serviceApi.ServiceToken("password","your cl
ient id", null, "CSDAdmin", "CSDAdmin", null,"Internal");

            //Create a new Business Object api object and add the default h
eader
            var businessObjectApi = new BusinessObjectApi("http://your serv
er/CherwellApi/");
            businessObjectApi.Configuration.AddDefaultHeader("Authorization
","Bearer " + tokenResponse.AccessToken);

            //Create a new Searches api object and add the default header
            var searchesApi = new SearchesApi("http://your server/CherwellA
```

```
pi");
            searchesApi.Configuration.AddDefaultHeader("Authorization","Bea
rer " + tokenResponse.AccessToken);

            //Get the Business Object summary for customer internal
            var businessObjectSummaryCustomerInternal = businessObjectApi.B
usinessObjectGetBusinessObjectSummaryByNameV1("CustomerInternal");

            //Get the Business Object schema for customer internal
            var schemaResponse = businessObjectApi.BusinessObjectGetBusines
sObjectSchemaV1(businessObjectSummaryCustomerInternal[0].BusObId, false);

            //Create the Search results request to lookup the customer and
get the customers record ID
            var searchResultsRequest = new ApiTrebuchetWebApiDataContractsS
earchesSearchResultsRequest();
            searchResultsRequest.BusObId = businessObjectSummaryCustomerInt
ernal[0].BusObId;
            searchResultsRequest.Filters = new List<ApiTrebuchetWebApiDataC
ontractsSearchesFilterInfo>();
            var filterInfo = new ApiTrebuchetWebApiDataContractsSearchesFil
terInfo();
            filterInfo.FieldId = schemaResponse.FieldDefinitions.First(f =
> f.Name == "FullName").FieldId;
            filterInfo.Operator = "eq";
            filterInfo.Value = "Eric Cox";
            searchResultsRequest.Filters.Add(filterInfo);

            //Run the Search
            var searchResultsResponse = searchesApi.SearchesGetSearchResult
sAdHocV1(searchResultsRequest);

            //Set the record ID to be used in the creation of the Incident
            var customerRecId = searchResultsResponse.BusinessObjects[0].Bu
```

```
sObRecId;


            //Get the field template for Incident to help set the fields
            var templateRequest = new ApiTrebuchetWebApiDataContractsBusine
ssObjectTemplateRequest();

            //Get the Business Object summary for Incident
            var businessObjectSummaryIncident = businessObjectApi.Business
bjectGetBusinessObjectSummaryByNameV1("Incident");

            templateRequest.BusObId = businessObjectSummaryIncident[0].BusO
bId;
            templateRequest.IncludeAll = true;

            //Use the template to set the fields
            var templateResponse = businessObjectApi.BusinessObjectGetBusin
essObjectTemplateV1(templateRequest);

            SetFieldValue(templateResponse.Fields, "Status", "New");
            SetFieldValue(templateResponse.Fields, "Description", "New Inci
dent");
            SetFieldValue(templateResponse.Fields, "ShortDescription", "Sho
rt Description");
            SetFieldValue(templateResponse.Fields, "CustomerRecID", custome
rRecId);
            SetFieldValue(templateResponse.Fields, "Priority", "2");
            SetFieldValue(templateResponse.Fields, "Source", "Phone");
            SetFieldValue(templateResponse.Fields, "IncidentType", "Inciden
t");
            SetFieldValue(templateResponse.Fields, "Service", "Employee Sup
port");
            SetFieldValue(templateResponse.Fields, "Category", "Add/Change"
);
```

```
            SetFieldValue(templateResponse.Fields, "Subcategory", "New Empl
oyee Setup");

            //Create the save request
            var saveRequest = new ApiTrebuchetWebApiDataContractsBusinessOb
jectSaveRequest();
            saveRequest.BusObId = businessObjectSummaryIncident[0].BusObId;
            saveRequest.Fields = templateResponse.Fields;

            //Create the Incident
            var saveResponse = businessObjectApi.BusinessObjectSaveBusiness
ObjectV1(saveRequest);

        }

        public void SetFieldValue(List<ApiTrebuchetWebApiDataContractsBusin
essObjectFieldTemplateItem> fields, string fieldName,
            string fieldValue)
        {
            var fieldTemplate = fields.First(s => s.Name.Equals(fieldName))
;
            if (fieldTemplate != null)
            {
                fieldTemplate.Value = fieldValue;
                fieldTemplate.Dirty = true;
            }
        }
    }
}
```

Related tasks

Using Swagger Code Generation

PowerShell Example: Create an Incident

The following example shows how to create an Incident using PowerShell.

```
The following example shows how to create an Incident using PowerShell.

#------------------------------------------------------------------------
-----
# Functions
#------------------------------------------------------------------------
-----

Function Set-FieldValue
{
    [CmdletBinding()]
    Param(
        [Parameter(Position=0, Mandatory=$True)]
            [PSCustomObject]$template
        ,[Parameter(Position=0, Mandatory=$True)]
            [String]$fieldName
        ,[Parameter(Position=0, Mandatory=$True)]
            [String]$value
    )

    $field = $template.fields | Where-Object {$_.name -eq $fieldName}

    if (!$field)
    {
        throw [System.Exception]"fieldName does not exist in template"
    }

    $field.value = $value
    $field.dirty = $true

}
```

```
#--------------------------------------------------------------------------
-----
# Logon to Service
#--------------------------------------------------------------------------
-----

# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "https://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }



#--------------------------------------------------------------------------
```

```
-----
# Get Customer Data
#-------------------------------------------------------------------------
-----

# Get the business object summary for customer internal
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/Custome
rInternal"
$summaryResults = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentTy
pe application/json -Header $requestHeader
$busobId = $summaryResults[0].busobId

# Get the business object schema for customer interal
$schemaUri = $baseUri + "api/V1/getbusinessobjectschema/busobid/" + $busobI
d
$schemaResults = Invoke-RestMethod -Method GET -Uri $schemaUri -ContentTyp
e application/json -Header $requestHeader

# Get the fieldId for the Full Name field so we can use it for a search.
$fullNameField = $schemaResults.fieldDefinitions | Where-Object {$_.display
Name -eq "Full name"}

# Create the search results request to lookup the customer and get the cust
omers recid
$filterInfo =
@{
        fieldId = $($fullNameField.fieldId);
        operator = "eq";
        value = "Eric Cox"
}
$searchResultsRequest =
@{
        busObID = $busobId;
        filters = @($filterInfo)
```

```
} | ConvertTo-Json

# Run the search
$searchUri = $baseUri + "api/V1/getsearchresults"
$searchResponse = Invoke-RestMethod -Method POST -Uri $searchUri -ContentTy
pe application/json -Header $requestHeader -Body $searchResultsRequest

# Set the recid to be used in the creation of the incident
$customerRecId = $searchResponse.businessObjects[0].busObRecid

# Get the business object summary for incident
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/Inciden
t"
$summaryResponse = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentT
ype application/json -Header $requestHeader
$busobId = $summaryResponse[0].busobId



#-------------------------------------------------------------------------
-----
# Create a Business Object Template for BusObID for the specified criteria
#-------------------------------------------------------------------------
-----


# Create request for the business object template POST method
$getTemplateUri = $baseUri + "api/V1/GetBusinessObjectTemplate"
$templateRequest =
@{
    busObId = $busobId;
    includeRequired = $true;
    includeAll = $true
} | ConvertTo-Json
```

```
$templateResponse = Invoke-RestMethod -Method POST -Uri $getTemplateUri -He
ader $requestHeader -ContentType application/json -Body $templateRequest

# Set values in the template
Set-FieldValue -template $templateResponse -fieldName "Status" -value "New"
Set-FieldValue -template $templateResponse -fieldName "Description" -value
"From Powershell using REST API"
Set-FieldValue -template $templateResponse -fieldName "ShortDescription" -v
alue "New Incident"
Set-FieldValue -template $templateResponse -fieldName "CustomerRecID" $cust
omerRecId
Set-FieldValue -template $templateResponse -fieldName "Priority" -value "2"
Set-FieldValue -template $templateResponse -fieldName "Source" -value "Phon
e"
Set-FieldValue -template $templateResponse -fieldName "IncidentType" -valu
e "Incident"
Set-FieldValue -template $templateResponse -fieldName "Service" -value "Emp
loyee Support"
Set-FieldValue -template $templateResponse -fieldName "Category" -value "Ad
d/Change"
Set-FieldValue -template $templateResponse -fieldName "Subcategory" -value
"New Employee Setup"



#-------------------------------------------------------------------------
-----
# Create New Business Object
#-------------------------------------------------------------------------
-----

# Get the fields portion of the template and use it in the request for a ne
w BO
$createBOUri = $baseuri + "api/V1/SaveBusinessObject"
```

```
$createBORequest =
@{
    busObId = $busobId;
    fields = @($($templateResponse.fields))
} | ConvertTo-Json

# Submit business object to server
$createBOResponse = Invoke-RestMethod -Method POST -Uri $createBOUri -Heade
r $requestHeader -ContentType application/json -Body $createBORequest
```

C# Example: Upload Attachments

The following example shows how to upload attachments using C#.

Note: Run Swagger Code Generation before attempting to modify the code so that you do not receive reference errors.

```
using System.IO;
using IO.Swagger.Api;

namespace Trebuchet.WebApi.IntegrationTests.ExamplesForCustomers.Attachment
s
{
    public class UploadAttachment
    {
        public void UploadAnAttachmentByBusinessObjectIdAndRecordId()
        {
            //Get an access token using CSM credentials
            var serviceApi = new ServiceApi("https://your server/CherwellAp
i/");
            var tokenResponse = serviceApi.ServiceToken("password","your cl
ient id", null, "CSDAdmin", "CSDAdmin", null,"Internal");

            //Create a new Business Object api object and add the default h
eader
            var businessObjectApi = new BusinessObjectApi("https://your ser
ver/CherwellApi/");
            businessObjectApi.Configuration.AddDefaultHeader("Authorization
","Bearer " + tokenResponse.AccessToken);

            //Get the Business Object summary for Incident
            var businessObjectSummaryIncident = businessObjectApi.BusinessO
bjectGetBusinessObjectSummaryByNameV1("Incident");

            var allFileBytes = File.ReadAllBytes("path to file");
```

```
            var totalSize = allFileBytes.Length;

            businessObjectApi.BusinessObjectUploadBusinessObjectAttachmentB
yIdAndRecIdV1(allFileBytes,"TextFile", businessObjectSummaryIncident[0].Bus
ObId,"941c01f21ee2de2f65ace74428a5f5955a3e621314", 0,totalSize,string.Empty
,string.Empty);
        }
    }
}
```

Related tasks

Using Swagger Code Generation

PowerShell Example: Upload Attachments

The following example shows how to upload attachments using PowerShell.

```
# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "https://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Get the business object summary for Incident. This will give us the busOb
Id
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/Inciden
t"
$summaryResults = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentTy
pe application/json -Header $requestHeader
$busObId = $summaryResults[0].busobId
```

```
# Get an incident rec id from the public id
$getIncidentUri = $baseUri + "api/V1/getbusinessobject/busobid/${busobid}/p
ublicid/102521"
$getIncidentResponse = Invoke-RestMethod -Method GET -Uri $getIncidentUri -
ContentType application/json -Header $requestHeader
$busObRecId = $getIncidentResponse.busObRecId

# Create a text file
"This file was attached to this incident by the API example for PowerShell
" | Out-File -FilePath "$($env:TEMP)\test.txt"
$file = Get-ChildItem -Path "$($env:TEMP)\test.txt"
$filename = $file.Name
$totalsize = $file.Length
$allFileBytes = [System.IO.File]::ReadAllBytes($file.FullName)
$offset = 0

# Upload the file to the incident
$uploadAttachmentUri = $baseUri + "api/V1/uploadbusinessobjectattachment/fi
lename/${filename}/busobid/${busobid}/busobrecid/${busobrecid}/offset/${off
set}/totalsize/${totalsize}"
$uploadAttachmentRequest = $allFileBytes
$uploadAttachmentResponse = Invoke-RestMethod -Method POST -Uri $uploadAtta
chmentUri -ContentType application/json -Header $requestHeader -Body $uploa
dAttachmentRequest

# Delete the text file that was created
Remove-Item $file.FullName -Force
```

C# Example: Delete a Business Object

The following example shows how to delete a Business Object using C#.

Note: Run Swagger Code Generation before attempting to modify the code so that you do not receive reference errors.

```
using IO.Swagger.Api;

namespace Trebuchet.WebApi.IntegrationTests.ExamplesForCustomers.Delete
{
    public class DeleteABusinessObject
    {
        public void DeleteABusinessObjectByPublicId()
        {
            //Get an access token using CSM credentials
            var serviceApi = new ServiceApi("https://your server/CherwellAp
i/");
            var tokenResponse = serviceApi.ServiceToken("password","your cl
ient id", null, "CSDAdmin", "CSDAdmin", null,"Internal");

            //Create a new Business Object api object and add the default h
eader
            var businessObjectApi = new BusinessObjectApi("https://your ser
ver/CherwellApi/");
            businessObjectApi.Configuration.AddDefaultHeader("Authorization
","Bearer " + tokenResponse.AccessToken);

            //Get the Business Object summary by name.  This returns the Bu
siness Object ID
            var businessObjectSummaryByName = businessObjectApi.BusinessObj
ectGetBusinessObjectSummaryByNameV1("Incident");

            //Delete Business Object
            businessObjectApi.BusinessObjectDeleteBusinessObjectByPublicIdV
```

```
1(businessObjectSummaryByName[0].BusObId, "123456");
        }
    }
}
```

Related tasks

Using Swagger Code Generation

PowerShell Example: Delete A Business Object

The following example shows how to delete a Business Object using PowerShell.

```
# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "https://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Get the business object summary for Incident. This will give us the busOb
Id
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/Inciden
t"
$summaryResults = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentTy
pe application/json -Header $requestHeader
$busObId = $summaryResults[0].busobId
```

```
# Delete business object
$deleteIncidentUri = $baseUri + "api/V1/deletebusinessobject/busobid/${buso
bid}/publicid/102522"
$deleteIncidentResponse = Invoke-RestMethod -Method DELETE -Uri $deleteInci
dentUri -ContentType application/json -Header $requestHeader
```

Using Search Operations

Search operations support getting search data from either a saved query or through an ad-hoc approach. An ad-hoc search is not based on a existing saved search. You can specify filters, columns, general search, and sorting capabilities. At a minimum, you must specify the Business Object ID.

Usage: Filters

To specify filtering, use the Filters member in the SearchResultsRequest. The Filters member is a collection of FilterInfo data structures. A FilterInfo contains the full field ID, operator and value.

You can specify more than one filter. If you add multiple filters for the same field ID, the result is an OR operation between those filters. If the field IDs are different, the result is an AND operation between those filters.

The following table shows the supported operators.

eq	Equals specified value
gt	Greater than specified value
lt	Less than specified value
contains	Contains specified value
startswith	Starts with specified value

The following example shows how to specify a filter on the Incident ID field to find a specific incident.

```
ApiTrebuchetWebApiDataContractsSearchesSearchResultsRequest searchResults
= new ApiTrebuchetWebApiDataContractsSearchesSearchResultsRequest();
searchResults.BusObId = "6dd53665c0c24cab86870a21cf6434ae";

// Create a filter on the incident ID field
searchResults.Filters = new System.Collections.Generic.List<ApiTrebuchetWeb
ApiDataContractsSearchesFilterInfo>();

ApiTrebuchetWebApiDataContractsSearchesFilterInfo filter1 = new ApiTrebuche
tWebApiDataContractsSearchesFilterInfo();
filter1.FieldId = "BO:6dd53665c0c24cab86870a21cf6434ae,FI:6ae282c55e8e4266a
e66ffc070c17fa3";
filter1.Operator = "eq";
filter1.Value = "100216";
searchResults.Filters.Add(filter1);
```

```
SearchesApi searches = new SearchesApi(apiClient);
ApiTrebuchetWebApiDataContractsSearchesSearchResultsResponse response = sea
rches.SearchesGetSearchResultsAdHocV1(searchResults);
```

In the following example, there are two filters for the incident ID field. This means that any incidents with IncidentID = 100216 OR IncidentID=200367 will be found.

```
ApiTrebuchetWebApiDataContractsSearchesSearchResultsRequest searchResults
= new ApiTrebuchetWebApiDataContractsSearchesSearchResultsRequest();
searchResults.BusObId = "6dd53665c0c24cab86870a21cf6434ae";

// Create a filter on the incident ID field
searchResults.Filters = new System.Collections.Generic.List<ApiTrebuchetWeb
ApiDataContractsSearchesFilterInfo>();

ApiTrebuchetWebApiDataContractsSearchesFilterInfo filter1 = new ApiTrebuche
tWebApiDataContractsSearchesFilterInfo();
filter1.FieldId = "BO:6dd53665c0c24cab86870a21cf6434ae,FI:6ae282c55e8e4266a
e66ffc070c17fa3";
filter1.Operator = "eq";
filter1.Value = "100216";
searchResults.Filters.Add(filter1);

ApiTrebuchetWebApiDataContractsSearchesFilterInfo filter2 = new ApiTrebuche
tWebApiDataContractsSearchesFilterInfo();
filter2.FieldId = "BO:6dd53665c0c24cab86870a21cf6434ae,FI:6ae282c55e8e4266a
e66ffc070c17fa3";
filter2.Operator = "eq";
filter2.Value = "200367";
searchResults.Filters.Add(filter2);

// Query for data
SearchesApi searches = new SearchesApi(apiClient);
ApiTrebuchetWebApiDataContractsSearchesSearchResultsResponse response = sea
```

```
rches.SearchesGetSearchResultsAdHocV1(searchResults)
;
```

Usage: Fields

By default, if you do not specify fields, a default set of fields (defined by the default Grid definition for the Business Object) is used to determine which fields will be displayed. You can override this behavior and specify the fields to include in your results. Use the Fields collection and add the full IDs of the fields to include:

```
ApiTrebuchetWebApiDataContractsSearchesSearchResultsRequest searchResults
= new ApiTrebuchetWebApiDataContractsSearchesSearchResultsRequest();
searchResults.BusObId = "6dd53665c0c24cab86870a21cf6434ae";

searchResults.Fields= new System.Collections.Generic.List<string>();
searchResults.Fields.Add("BO:6dd53665c0c24cab86870a21cf6434ae,FI:6ae282c55e
8e4266ae66ffc070c17fa3");
```

Usage: Search Results Field Templates

Use *POST api/V1/getsearchresultsbusinessobjects* to use the powerful filtering, sorting, and field specification abilities of the search results API to produce results that are in the field template data structure syntax (see Business Objects / Field Templates).

You can then update values within that set of field template and then update those records. There are several scenarios where this might be useful. For example, you can find all incidents older than 1 year, and then set their status to closed.

C# Example: Get Search Items

The following example shows how to get search items using C#.

Note: Run Swagger Code Generation before attempting to modify the code so that you do not receive reference errors.

```
using IO.Swagger.Api;


namespace Trebuchet.WebApi.IntegrationTests.ExamplesForCustomers.Searches
{

    public class GetSearchItems
    {

        public void GetSearchItemsByAssociationAndScope()
        {
            //Get an access token using CSM credentials
            var serviceApi = new ServiceApi("http://server/CherwellApi/");
            var tokenResponse = serviceApi.ServiceToken("password", "your c
lient id", null,"CSDAdmin", "CSDAdmin", null, "Internal");

            //Create a new Searches api object and add the default header
            var searchesApi = new SearchesApi("http://server/CherwellApi/")
;
            searchesApi.Configuration.AddDefaultHeader("Authorization","Bea
rer " + tokenResponse.AccessToken);

            //Create a new Business Object api object and add the default h
eader
            var businessObjectApi = new BusinessObjectApi("http://server/Ch
erwellApi/");
            businessObjectApi.Configuration.AddDefaultHeader("Authorization
","Bearer " + tokenResponse.AccessToken);
```

```
            //Get the Business Object summary by name.  This returns the Bu
siness Object ID
            var businessObjectSummaryByName =  businessObjectApi.BusinessOb
jectGetBusinessObjectSummaryByNameV1("Incident");

            var searchItemResponse = searchesApi.SearchesGetSearchItemsByAs
sociationScopeV1(businessObjectSummaryByName[0].BusObId, "Global", null);
        }
    }
}
```

Related tasks

Using Swagger Code Generation

PowerShell Example: Get Search Items

The following example shows how to get search items using PowerShell.

```
# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "http://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Get the business object summary for Incident. This will give us the busOb
Id
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/Inciden
t"
$summaryResults = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentTy
pe application/json -Header $requestHeader
$busObId = $summaryResults[0].busobId
```

```
# Get the search items (saved queries) with association of Incident from th
e Global scope
$searchUri = $baseUri + "api/V1/getsearchitems/association/${busObId}/scope
/Global"
$searchResponse = Invoke-RestMethod -Method GET -Uri $searchUri -ContentTyp
e application/json -Header $requestHeader
```

C# Example: Perform an Ad-hoc Search with Filter

The following example shows how to perform an ad-hoc search for Incidents using a filter.

Note: Run Swagger Code Generation before attempting to modify the code so that you do not receive reference errors.

```
using System.Collections.Generic;
using System.Linq;
using IO.Swagger.Api;
using IO.Swagger.Model;

namespace Trebuchet.WebApi.IntegrationTests.ExamplesForCustomers.Searches
{
    public class AdHocSearchWithFilter
    {
        public void SearchResultsWithFilter()
        {
                                                                            /
/Get an access token using CSM credentials
            var serviceApi = new ServiceApi("http://your server/CherwellApi
/");
            var tokenResponse = serviceApi.ServiceToken("password","your cl
ient id", null, "CSDAdmin", "CSDAdmin", null,"Internal");

            //Create a new Business Object api object and add the default h
eader
            var businessObjectApi = new BusinessObjectApi("http://your serv
er/CherwellApi/");
            businessObjectApi.Configuration.AddDefaultHeader("Authorization
","Bearer " + tokenResponse.AccessToken);

            //Create a new Searches api object and add the default header
            var searchesApi = new SearchesApi("http://your server/CherwellA
pi/");
```

```
            searchesApi.Configuration.AddDefaultHeader("Authorization","Bea
rer " + tokenResponse.AccessToken);

            //Get the Business Object summary by name.  This returns the Bu
siness Object ID
            var businessObjectSummaryByName = businessObjectApi.BusinessObj
ectGetBusinessObjectSummaryByNameV1("Incident");

            //Get the Incident Business Object template
            var templateRequest = new ApiTrebuchetWebApiDataContractsBusine
ssObjectTemplateRequest
            {
                BusObId = businessObjectSummaryByName[0].BusObId
            };
            var templateRespsonse = businessObjectApi.BusinessObjectGetBusi
nessObjectTemplateV1(templateRequest);

            //Create Search results request
            var searchResults = new ApiTrebuchetWebApiDataContractsSearches
SearchResultsRequest();
            searchResults.BusObId = businessObjectSummaryByName[0].BusObId;

            //Create a filter on the Incident ID field
            searchResults.Filters = new List<ApiTrebuchetWebApiDataContract
sSearchesFilterInfo>();

            var filter = new ApiTrebuchetWebApiDataContractsSearchesFilterI
nfo();
            filter.FieldId = templateRespsonse.Fields.First(f => f.Name ==
"ShortDescription").FieldId;
            filter.Operator = "eq";
            filter.Value = "Down System";
            searchResults.Filters.Add(filter);
```

```
            //Query for data
            var response = searchesApi.SearchesGetSearchResultsAdHocV1(sear
chResults);
        }
    }
}
```

Related tasks

Using Swagger Code Generation

PowerShell Example: Perform an Ad-hoc Search with Filter

The following example shows how to perform an ad-hoc search for Incidents using a filter.

```
# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "http://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Get the business object summary for Incident. This will give us the busOb
Id
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/Inciden
t"
$summaryResponse = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentT
ype application/json -Header $requestHeader
```

```
$busObId = $summaryResponse[0].busobId

# Get the business object template for Incident. We need this to get the fi
eld id for ShortDescription
$templateUri = $baseUri + "api/V1/getbusinessobjecttemplate"
$templateRequestBody =
@{
    busObId = $busobId
    includeAll = $true
} | ConvertTo-Json
$templateResponse = Invoke-RestMethod -Method POST -Uri $templateUri -Conte
ntType application/json -Header $requestHeader -Body $templateRequestBody
$shortDescField = $templateResponse.fields | Where-Object {$_.name -eq "Sho
rtDescription"}

# Put together the request
$searchResultsRequest =
@{      Q
    BusObId = $busobId;
    Filters = @(
        @{
            fieldId = $shortDescField.fieldId;
            Operator = "eq";
            Value = "Printer Issue"
        })
} | ConvertTo-Json

# Run the search
$searchUri = $baseUri + "api/V1/getsearchresults"
$searchResultsResponse = Invoke-RestMethod -Method POST -Uri $searchUri -Co
ntentType application/json -Header $requestHeader -Body $searchResultsReque
st

Write-Host $searchResultsResponse
```

C# Example: Perform an Ad-hoc Search with Sorting

The following example shows how to perform an ad-hoc search for Incidents sorted in ascending order.

Note: Run Swagger Code Generation before attempting to modify the code so that you do not receive reference errors.

```
using System;
using System.Collections.Generic;
using System.Linq;
using IO.Swagger.Api;
using IO.Swagger.Model;

namespace Trebuchet.WebApi.IntegrationTests.ExamplesForCustomers.Searches
{
    public class AdHocSearchAscending
    {
        public void SearchResultsSortingAscendingByIncidentStatus()
        {
            //Get an access token using CSM credentials
            var serviceApi = new ServiceApi("http://your server/CherwellApi
/");
            var tokenResponse = serviceApi.ServiceToken("password","your cl
ient id", null, "CSDAdmin", "CSDAdmin", null,"Internal");

            //Create a new Business Object api object and add the default h
eader
            var businessObjectApi = new BusinessObjectApi("http://your serv
er/CherwellApi/");
            businessObjectApi.Configuration.AddDefaultHeader("Authorization
","Bearer " + tokenResponse.AccessToken);

            //Create a new Searches api object and add the default header
```

```
            var searchesApi = new SearchesApi("http://your server/CherwellA
pi/");
            businessObjectApi.Configuration.AddDefaultHeader("Authorization
","Bearer " + tokenResponse.AccessToken);

            //Get the Business Object summary by name.  This returns the Bu
siness Object ID
            var businessObjectSummaryByName = businessObjectApi.BusinessObj
ectGetBusinessObjectSummaryByNameV1("Incident");

            //Get the schemasresponse for Incident to get the field ID for
Incident ID
            var schemaResponse = businessObjectApi.BusinessObjectGetBusines
sObjectSchemaV1(businessObjectSummaryByName[0].BusObId);

            //Put together the request
            var searchResultsRequest = new ApiTrebuchetWebApiDataContractsS
earchesSearchResultsRequest
            {
                BusObId = businessObjectSummaryByName[0].BusObId,
                PageSize = 200,
                Sorting = new List<ApiTrebuchetWebApiDataContractsSearchesS
ortInfo>
                {
                    new ApiTrebuchetWebApiDataContractsSearchesSortInfo
                    {
                        FieldId =
                            schemaResponse.FieldDefinitions.First(n => n.Na
me == "Status").FieldId,
                        SortDirection = 1
                    }
                }
            };
```

```
            //Run the Search
            searchesApi.SearchesGetSearchResultsAdHocV1(searchResultsReques
t);
        }
    }
}
```

Related tasks

Using Swagger Code Generation

PowerShell Example: Perform an Ad-hoc Search with Sorting

The following example shows how to perform an ad-hoc search for Incidents sorted in ascending order.

```
# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "http://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Get the business object summary for Incident. This will give us the busOb
Id
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/Inciden
t"
$summaryResponse = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentT
ype application/json -Header $requestHeader
```

```
$busObId = $summaryResponse[0].busobId

# Get the business object schema for Incident. We need this to get the fiel
d id for Status
$schemaUri = $baseUri + "api/V1/getbusinessobjectschema/busobid/" + $busobI
d
$schemaResponse = Invoke-RestMethod -Method GET -Uri $schemaUri -ContentTyp
e application/json -Header $requestHeader
$statusField = $schemaResponse.fieldDefinitions | Where-Object {$_.name -e
q "Status"}

# Put together the request
$searchResultsRequest =
@{
    BusObId = $busObId;
    PageSize = 200;
    Sorting = @(
        @{
            fieldId = $statusField.fieldId;
            sortDirection = 1
        })
} | ConvertTo-Json

# Run the search
$searchUri = $baseUri + "api/V1/getsearchresults"
$searchResultsResponse = Invoke-RestMethod -Method POST -Uri $searchUri -Co
ntentType application/json -Header $requestHeader -Body $searchResultsReque
st
```

Security Usage and Examples

Security operations enable you to perform CRUD (create, read, update and delete) operations for single User accounts or in a batch. You can also add and remove Users from Teams.

Usage: Creating and Updating Users

Use these operations to create or update users:

- saveuser
- saveuserbatch

To create users, do not include a record ID or public ID in the request.

To update existing users, specify the record ID or public ID in the request. If duplicate public IDs exist, you must use the record ID.

C# Example: Create a User

The following example shows how to create a CSM User account using C#.

Note: Run Swagger Code Generation before attempting to modify the code so that you do not receive reference errors.

```
using System;
using System.Collections.Generic;
using System.Linq;
using IO.Swagger.Api;
using IO.Swagger.Model;



namespace Trebuchet.WebApi.IntegrationTests.ExamplesForCustomers.Create_an_
Object
{
    public class CreateUser
    {
        public void CreateNewUser()
        {
            //Get an access token using CSM credentials
            var serviceApi = new ServiceApi("http://your server/CherwellApi
/");
            var tokenResponse = serviceApi.ServiceToken("password","your cl
ient id", null, "CSDAdmin", "CSDAdmin", null,"Internal");

            //Create a new Business Object api object and add the default h
eader
            var businessObjectApi = new BusinessObjectApi("http://your serv
er/CherwellApi/");
            businessObjectApi.Configuration.AddDefaultHeader("Authorization
","Bearer " + tokenResponse.AccessToken);

            //Create a new Security api object and add the default header
```

```
            var securityApi = new SecurityApi("http://your server/CherwellA
pi/");
            securityApi.Configuration.AddDefaultHeader("Authorization","Bea
rer " + tokenResponse.AccessToken);

            //Get the Business Object summary for user info
            var businessObjectSummaryUserInfo = businessObjectApi.Business0
bjectGetBusinessObjectSummaryByNameV1("UserInfo");

            //Get the Security Group information
            var securityGroupResponse = securityApi.SecurityGetSecurityGrou
psV1();

            //Get the template for userinfo.  Use this to get field ids
            var templateRequest = new ApiTrebuchetWebApiDataContractsBusine
ssObjectTemplateRequest
            {
                BusObId = businessObjectSummaryUserInfo[0].BusObId,
                IncludeAll = true
            };

            var templateResponse = businessObjectApi.BusinessObjectGetBusin
essObjectTemplateV1(templateRequest);

            //Create the user save request
            var userSaveRequest = new ApiTrebuchetWebApiDataContractsUsersU
serSaveRequest
            {
                AccountLocked = false,
                BusObId = businessObjectSummaryUserInfo[0].BusObId,
                DisplayName = "Test User",
                LdapRequired = false,
                LoginId = "Test",
                NextPasswordResetDate = null,
```

```
                Password = "P@ssword",
                PasswordNeverExpires = true,
                SecurityGroupId = securityGroupResponse.SecurityGroups.Firs
t(f => f.GroupName == "Admin").GroupId,
                UserCannotChangePassword = false,
                UserMustChangePasswordAtNextLogin = false,
                UserInfoFields = new List<ApiTrebuchetWebApiDataContractsBu
sinessObjectFieldTemplateItem>
                {
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "FullName",
                        Value = "Test User",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "FullName").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "EmployeeID",
                        Value = "123456",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "EmployeeID").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "Comments",
                        Value = "Created by api",
                        FieldId = templateResponse.Fields.First(f => f.Nam
```

```
e == "Comments").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "Department",
                        Value = "IT",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "Department").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "Office",
                        Value = "Colorado Springs",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "Office").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "Phone",
                        Value = "719-777-7777",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "Phone").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "CellPhone",
```

```
                        Value = "719-777-7778",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "CellPhone").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "FirstName",
                        Value = "Test",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "FirstName").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "MiddleInitial",
                        Value = "C",
                        FieldId =
                            templateResponse.Fields.First(f => f.Name == "M
iddleInitial").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "LastName",
                        Value = "User",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "LastName").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
```

```
                    {
                        Dirty = true,
                        Name = "HomePhone",
                        Value = "719-777-7779",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "HomePhone").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "Address",
                        Value = "1234 Cherwell Ave",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "Address").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "City",
                        Value = "Colorado Springs",
                        FieldId = templateResponse.Fields.First(f => f.Nam
e == "City").FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "ProvinceStateName",
                        Value = "CO",
                        FieldId =
                            templateResponse.Fields.First(f => f.Name == "P
rovinceStateName")
```

```
                                .FieldId
                    },
                    new ApiTrebuchetWebApiDataContractsBusinessObjectFieldT
emplateItem
                    {
                        Dirty = true,
                        Name = "PostalCodeZip",
                        Value = "80132",
                        FieldId =
                            templateResponse.Fields.First(f => f.Name == "P
ostalCodeZip").FieldId
                    }
                }
            };

            //Create the user
            var usersApi = new UsersApi("http://your server/CherwellApi/");
            usersApi.Configuration.AddDefaultHeader("Authorization","Beare
r " + tokenResponse.AccessToken);
            try
            {
                usersApi.UsersSaveUserV1(userSaveRequest);
            }
            catch (Exception ex)
            {
                Assert.Fail(ex.Message);
            }
        }
    }
}
```

Related tasks

Using Swagger Code Generation

PowerShell Example: Create a User

The following example shows how to create a CSM User account using PowerShell.

```
# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "http://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Get the business object summary for UserInfo. This will give us the busOb
Id
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/UserInf
o"
$summaryResults = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentTy
pe application/json -Header $requestHeader
$busObId = $summaryResults[0].busobId
```

```
# Get the security group information
$securityGroupUri = $baseUri + "api/V2/getsecuritygroups"
$securityGroupResponse = Invoke-RestMethod -Method GET -Uri $securityGroupU
ri -ContentType application/json -Header $requestHeader
$adminGroup = $securityGroupResponse.securityGroups | Where-Object {$_.grou
pName -eq "Admin"}

# Get the template for UserInfo. Use this to get field ids
$templateUri = $baseUri + "api/V1/getbusinessobjecttemplate"
$templateRequestBody =
@{
    busObId = $busObId
    includeAll = $true
} | ConvertTo-Json
$templateResponse = Invoke-RestMethod -Method POST -Uri $templateUri -Conte
ntType application/json -Header $requestHeader -Body $templateRequestBody

# Create the save request
$userSaveUri = $baseUri + "api/V2/saveuser"
$userSaveRequest =
@{
    AccountLocked = $false;
    BusObId = $busObId;
    DisplayName = "Test User";
    LdapRequired = $false;
    LoginId = "Test";
    NextPasswordResetDate = $null;
    Password = "P@ssword";
    PasswordNeverExpires = $true;
    SecurityGroupId = $adminGroup.groupId;
    UserCannotChangePassword = $false;
    UserMustChangePasswordAtNextLogin = $false;
    UserInfoFields = @(
```

```
            @{
                Dirty = $true;
                Name = "FullName";
                Value = "Test User";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "FullName"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "EmployeeID";
                Value = "123456";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "EmployeeID"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "Comments";
                Value = "Created by API in PowerShell";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "Comments"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "Department";
                Value = "IT";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "Department"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "Office";
                Value = "Colorado Springs";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "Office"}).fieldId
```

```
            };
            @{
                Dirty = $true;
                Name = "Phone";
                Value = "719-777-7777";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "Phone"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "CellPhone";
                Value = "719-777-7778";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "CellPhone"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "FirstName";
                Value = "Test";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "FirstName"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "MiddleInitial";
                Value = "C";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "MiddleInitial"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "LastName";
                Value = "User";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
```

```
e -eq "LastName"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "HomePhone";
                Value = "719-777-7779";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "HomePhone"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "Address";
                Value = "1234 Cherwell Ave";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "Address"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "City";
                Value = "Colorado Springs";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "City"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "ProvinceStateName";
                Value = "CO";
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "ProvinceStateName"}).fieldId
            };
            @{
                Dirty = $true;
                Name = "PostalCodeZip";
                Value = "80132";
```

```
                FieldId = ($templateResponse.fields | Where-Object {$_.nam
e -eq "PostalCodeZip"}).fieldId
            }
        )
} | ConvertTo-Json

# Create the user
$userSaveResponse = Invoke-RestMethod -Method POST -Uri $userSaveUri -Heade
r $requestHeader -ContentType application/json -Body $userSaveRequest
```

Using One-Step Action Operations

One-Step™ Action operations return a list of available One-Step Actions and support executing One-Step Actions through the API.

Usage: Identifying One-Step Actions

The *GET /api/V1/getonestepactions* operations provide lists of One-Step Actions. These lists can be filtered by the following classifications:

- Association
- Scope
- Scope Owner
- Folder

The lists generated from these calls include the standInKey that is a required parameter used to run One-Step Actions from the API.

Usage: Run a One-Step Action

The *GET /api/V1/runonestepaction* and *POST /api/V1/runonestepaction* operations allow you to run different types of One-Step Actions. There are three methods for running a One-Step Action from the REST API:

- Run a One-Step Action by Business Object record
- Run a stand-alone One-Step Action
- Run a One-Step Action that includes Prompts

To run a One-Step Action by Business Object record, the following parameters must be provided:

- standinkey
- busobid
- busobrecid

A stand-alone One-Step Action requires only the standInKey parameter. This parameter is included in the list returned by the *GET /api/V1/getonestepactions* operations. This type of method cannot be used if the One-Step Action includes prompts. Configure the One-Step Action so that there are no prompts to run it from this method.

To run a One-Step Action from the API that includes prompts, you must provide a request parameter that includes the prompt values. The request must be formatted according to the following request model.

Note: This request must include all Prompt values. If any Prompt values are missing, the response includes an error message that lists the missing values.

```
{
  "acquireLicense": true,
  "busObId": "string",
  "busObRecId": "string",
  "oneStepActionStandInKey": "string",
  "promptValues": [
    {
      "promptDefId": "string",
      "promptName": "string",
      "value": "string"
    }
  ]
}
```

The Prompt Def ID parameter is available by selecting the **Prompt ID** button on the Define Prompt dialog box in the One-Step Editor. Likewise, the Prompt Values parameter is available in properly formatted JSON by selecting the **Prompt Info** button on the first node of the One-Step Action in the One-Step Editor. See About Prompts for more information.

PowerShell Example: Get a List of One-Step Actions

This example shows how to get all global, Incident One-Step™ Actions using PowerShell.

```
# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "http://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Get the business object summary for the Incident. This will give us the b
usObId
$summaryUri = $baseUri + "api/V1/getbusinessobjectsummary/busobname/Inciden
t"
$summaryResults = Invoke-RestMethod -Method GET -Uri $summaryUri -ContentTy
pe application/json -Header $requestHeader
```

```
$busObId = $summaryResults[0].busobId

# Get the One-Step Actions with association of Incident from the Global sco
pe
$oneStepActionUri = $baseUri + "api/V1/getonestepactions/association/${busO
bId}/scope/Global"
$oneStepActionsResponse = Invoke-RestMethod -Method GET -Uri $oneStepAction
Uri -ContentType application/json -Header $requestHeader
```

PowerShell Example: Run a One-Step Action

These Powershell commands provide an example of running a simple, unassociated One-Step™ Action that creates a Business Object using a StandInKey.

```
# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "http://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Run a simple One-Step Action from the Global scope that requires no input
# Note the StandInKey is URL encoded, so ':' becomes '%3A' and '#' becomes
'%23'
$standInKey = "DefType%3AOneStepDef%23Scope%3AGlobal%23Id%3A9451b5b6d294dfe
8a1bb054dec9f4bbb1c07460a4a"
$oneStepActionUri = $baseUri + "api/V1/runonestepaction/standinkey/${standI
```

```
nKey}"
$oneStepActionsResponse = Invoke-RestMethod -Method GET -Uri $oneStepAction
Uri -ContentType application/json -Header $requestHeader
```

PowerShell Example: Run a One-Step Action that Includes a Prompt

These Powershell commands provide an example for running a One-Step™ Action using a OneStepActionRequest that provides an existing Incident and adds a Journal to it. In this One-Step Action, the contents are determined by a prompt.

```
# Set server login variables
$serverName = "your server"
$apiKey = "your client id"
$userName = "CSDAdmin"
$password = "CSDAdmin"
$baseUri = "http://${serverName}/CherwellAPI/"

# Get an access token
$tokenUri = $baseUri + "token"
$authMode = "Internal"
$tokenRequestBody =
@{
    "Accept" = "application/json";
    "grant_type" = "password";
    "client_id" = $apiKey;
    "username" = $userName;
    "password"= $password
}
$tokenResponse = Invoke-RestMethod -Method POST -Uri "${tokenUri}?auth_mode
=${authMode}&api_key=${apiKey}" -Body $tokenRequestBody

$requestHeader = @{ Authorization = "Bearer $($tokenResponse.access_token)
" }

# Build a One-Step Action Request that will run an Incident One-Step Action
# from the Global scope that adds a Journal to a specific Incident with the
# details for the Journal being provided by a prompt
$incidentBusObId = "6dd53665c0c24cab86870a21cf6434ae"
```

```
$recordId = "9451a88b3582418263efdf4fe0a5709443f8176be6"
$oneStepId = "9451b5d71fba1b00c809b5477691ebcde8f49786cd"
$promptInput = "This journal was created by the REST API"
# Either the promptDefId or the promptName can be used to provide a value f
or
# the prompt. If both are provided, it will match based on the ID.
$oneStepActionRequestBody = @"
{
  "acquireLicense": "true",
  "busObId": "${incidentBusObId}",
  "busObRecId": "${recordId}",
  "oneStepActionStandInKey": "DefType:OneStepDef#Scope:Global#Id:${oneStepI
d}#Owner:${incidentBusObId}",
  "promptValues": [
    {
      "promptDefId": "9451b5d880b9dd6497188d45d8a91fa1d9dad4856b",
      "promptName": "JournalDetailsPrompt",
      "value": "$promptInput"
    }
  ]
}
"@

$oneStepActionUri = ${baseUri} + "api/V1/runonestepaction"
$oneStepActionResponse = Invoke-RestMethod -Method POST -Uri $oneStepAction
Uri -Header $requestHeader -ContentType "application/json" -Body $oneStepAc
tionRequestBody
```

Operation Versioning

Cherwell REST API operation versioning is possible with these operations:

1. Add the operation version in the URL. (Example: http://localhost/CherwellRESTAPI/api/v1/serviceinfo)
2. Add the operation version in a custom header attribute. add x-api-version header to the request specifying the operation version number. (Example: http://localhost/CherwellRESTAPI/api/serviceinfo with a header x-api-version: 1)
3. Add a value to the Accept header. (Example: http://localhost/CherwellRESTAPI/api/serviceinfo with the Accept header application/json ; api-version=1)
4. Add a query string parameter to the URL. (Example: http://localhost/CherwellRESTAPI/api/serviceinfo?v=1)

The operation version of the Cherwell REST API differs from the underlying CSM version, though an installation is dependent on a version of CSM.

Operations will increment versions when request/response contracts are changed by removing properties, changing property data types, or changing the schema of complex data types of properties.

An operation version of an operation remains the same as long as there are no breaking changes to the request/response contracts.

The operation version will not be incremented for each CSM release unless breaking changes were made. New operations will be added as version 1 of those operations. If no version is specified, v1 is used. If a version is in the URL and that does not exist, the operation will not be found. If the version is in the URL, it will override all the other version specifying operations. For example, if operation 2, 3, or 4 are used, the URL cannot have a version segment.

Errors

The Cherwell REST API uses conventional HTTP response codes to indicate the success or failure of an API request.

In general:

- Codes in the 200 range indicate success.
- Codes in the 400 range indicate a failure error based on the information provided.

Here are a few http codes that you might see:

200	OK. Everything worked as expected.
400	Bad request, often due to a missing required parameter.
402	Request failed. Typically, the parameters are valid but the request failed.
404	Not found. The requested resource does not exist.

Not all errors map cleanly to HTTP response codes. In these cases, check the error message for more information.

For example, if you specify an incorrect client ID, you will get the following response:

```
Status Code 400
"{\"error\":\"invalid_client_id\",\"error_description\":\"Client 12345 is n
ot registered in the system.\"}"
```

If you supply invalid credentials, you will get the following response:

```
Status Code 400
"{\"error\":\"invalid_grant\",\"error_description\":\
"Could not login to system with the provided user ID and password.\"}"
```

Using Swagger Code Generation

Use swagger-codegen to generate client code.

Important: Changes to the Cherwell REST API may require you to regenerate Swagger code after you upgrade to a new version of CSM. If you receive errors in your API calls after you upgrade, follow the guidance in this article to regenerate client code.

Follow this process:

1. Download the swagger-codegen JAR from the Maven website.
2. Create a .bat file similar to the following example:

```
java -jar modules/swagger-codegen-cli/target/swagger-codegen-cli.jar ge
nerate -i
http://localhost/CherwellApi/swagger/docs/all -l csharp -o cherwell/cli
ent/csharp
```

3. Place the .bat file in the same level as the Swagger code gen modules directory.

Note: The example script creates output in a cherwell/client/csharp directory at the same level where the .bat file is placed.

4. Add the generated code from the output directory to your project and reference it.

You can now call the generated Swagger code for CSM.

Sample Authenticated Request

```
var service = new ServiceApi(baseuri);
var tokenResponse = service.ServiceToken("password", clientid, "", username
, password, "", "Internal");
var searchesApi = new SearchesApi(Baseuri);
searchesApi.DefaultHeader.Add("Authorization", "Bearer " + tokenResponse.Ac
cessToken);
var searchItemResponse = searchesApi.SearchesGetSearchItemsV1(null);
```

Parameter Glossary

Use the operation usage information for parameters for the Cherwell REST API.

The following conventions are used in this glossary:

- **Internal ID**: The database ID for a specific object.
- **Name**: Typically, the display name for an object.
- **Public ID**: Typically, the ID shown to users, such as an incident ID or a user's full name.
- **Record ID**: The database ID for a specific record.
- **Search Item**: Saved searches, also known as a stored query.
- **Search Results**: A set of records returned by a search.

Business Object attachment operations (requests or responses)

Parameter	Operation Usage
allfields	A flag to include all related Business Object fields.
attachBusObId	The internal ID for the type of Business Object to attach to another Business Object.
attachBusObName	The display name for the type of Business Object you want to attach to another Business Object.
attachBusObPublicId	The public ID for the type of Business Object you want to attach to another Business Object.
attachBusObRecId	The internal ID for the Business Object to attach to another Business Object.
attachedBusObId	The internal ID for the type of Business Object attached to another Business Object.
attachedBusObRecId	The internal ID for the Business Object attached to another Business Object.
attachmentFileId	The internal ID for a file attachment.
attachmentFileName	The path and file name of a file attachment.
attachmentFileType	The type of file attachment.
attachmentId	The internal ID for a record that contains information about a file attachment.
attachments	A list of objects related to Business Object attachments.
attachmentTypes	The type of file attachment: • 0 = Imported into the database. • 1 = Linked to an external file. • 2 = URL attachment.
body	The body of the request is the byte array of the file part being uploaded.

Parameter	Operation Usage
created	The date and time an attachment was added to a Business Object.
customSeparator	A custom separator for exports to a text file.
filename	The name of a file attachment being uploaded.
offset	The starting index of the file part being uploaded. For the first part, the offset should be zero.
totalsize	The size of a file, in bytes.
uncFilePath	A file path to a Business Object linked attachment.

Business Object operations (requests, responses, schema, or template requests)

Parameter	Operation Usage
allfields	A flag to include all related Business Object fields.
customGridId	The internal ID for the grid used to override field list settings.
fieldNames	A comma-delimited list of field names. By default, all fields are requested.
firstRecIdField	The first Record ID field specified for a Business Object.
foredit	For form operations, a flag to get an editable version of a form.
group	A flag to indicate that the Business Object is a group leader.
groupSummaries	A list of objects for members of a Business Object group.
html	The placeholder for the HTML field value. It is only populated on getting field values for rich text fields.
includeAll	A flag to include all fields with Business Object templates.
includerelationships	A flag to include schemas for related Business Objects. Default is false.
includeRequired	A flag to request all required fields with Business Object templates.
lookup	A flag indicating a Business Object type of Lookup.
major	A flag indicating a Business Object type of Major.
publicid/publicId	The public ID used to identify Business Object records.
relatedBusinessObjects	A list of objects for related Business Objects.
stateFieldId	The internal ID for the State field defined for a Business Object life cycle.
states	The display names for State values defined for a Business Object life cycle.
supporting	A flag indicating a Business Object type of Supporting.

Field schema responses

Parameter	Operation Usage
autoFill	A flag that indicates if auto-populate is enabled.
calculated	A flag that indicates if an expression is used to calculate the field's value.
caption	The title for a column on the Business Object grid.
category	A string that indicates the folder used to organize the field.
currencyCulture	The Currency Culture set for Number fields.
currencySymbol	The currency symbol for the Currency Culture set for Number fields
decimalDigits	The number of digits for a Number field that appear in the field value after the decimal point.
details	A string of attributes set for a field.
enabled	An expression that is being evaluated.
fieldDefinitions	A list of properties for each field.
hasDate	Indicates if a Date/Time field displays dates.
hasDefaultSortField	A flag to indicate if a field has a default sort order. Use with *defaultSortOrder*.
hasTime	Indicates if a Date/Time field displays times.
isBinary	A flag to indicate that field contains Binary data, such as an image.
isCurrency	A flag to indicate that a Number field stores currency values.
isDateTime	A flag to indicate a Date/Time field.
isFilterAllowed	A flag to indicate that filtering is allowed.
isFullTextSearchable	A flag to indicate if full-text searching is enabled for a field.
isLogical	A flag to indicate a Logical field.
isNumber	A flag to indicate a Number field.
isShortDate	A flag to indicate that a Date/Time field shows dates only.
isVisible	A flag to indicate if a field is visible.
maximumSize	For text fields, the maximum number of characters allowed.
readOnly	A flag to indicate if the field is read only (true) or editable (false).
sortable	A flag to indicate that a field is sortable (true) or not sortable (false).
sortOrder	Indicates sort order (ascending or descending).
storageName	The database name used for a field.
validated	A flag to indicate whether validation is defined for the field. See *details* for validation information.

Parameter	Operation Usage
wholeDigits	Indicates the number of whole digits specified for number fields.

Flags

Parameter	Operation Usage
dirty	A flag to update a field value if true. If false, the value is not updated.
hasError	A flag to indicate an error needs to be reviewed.
includelinks	A flag to request hyperlinks in results. Default is false.
links	For requests, a flag to request hyperlinks in results. Default is false. For responses, a list of links, including name and URL.
multiline	For prompts, a flag to indicate if a text prompt can contain multiple lines.
required	A flag to indicate an item is required (true) or optional (false).
saveRequests	A list of saveRequest objects that will be processed.
stopOnError	A flag to stop or continue on error.
usedefaultgrid	A flag to trigger the use of the related Business Objects default grid for the list of fields to return.

Internal IDs or names

Parameter	Operation Usage
association	The internal ID for the Business Object association.
busobid/busObID	The internal ID for a Business Object type, such as Incident or Task.
busobrecid	The internal ID for a single Business Object record.
fieldId	The internal ID for a field.
folder/folderId	The name or internal ID of an item's folder.
formid	The internal ID for a form.
gridDefinitions	The internal IDs for grids associated with a Business Object.
gridid	The internal ID for the custom grid that contains the field list.
name	The display name for an object.
owner	The internal ID for the Business Object association.
parentbusobid	The internal ID for the parent Business Object type.
scope	The name or internal ID of an item's scope. For example, *Global* and *Team* are scopes.

Parameter	Operation Usage
scopeowner	The internal ID or name of the scope owner. For example, if *Team* is the scope, *1st Level Support* might be the scope owner.
userrecordid/ userRecID	The internal ID for a CSM user.

Mobile form operations (responses)

Parameter	Operation Usage
actions	A list of action objects available for a form.
altitude	The altitude for a specific Business Object record (if location awareness is enabled).
galleryImage	The image shown on the form.
id	The internal ID for actions available for the form.
label	The display name for fields.
latitude	The latitude for a specific Business Object record (if location awareness is enabled).
locationInformation	A list of location objects for a specific Business Object record.
longitude	The longitude for a specific Business Object record (if location awareness is enabled).
multiline	A flag to indicate if a field can span multiple lines.
sectionFields	A list of fields and attributes for each section on a form.
sections	A list of sections available on a form.
targetBusObId	The Business Object ID for the Business Object type.
targetBusObRecId	The Business Object record ID.
title	The display name of a Business Object.

Queue operations (requests or responses)

Parameter	Operation Usage
historyNotes	Notes to be added to the history log for the item.
historyRecId	The record ID of the Business Object with the history.
historyText	The text of the history record for the item.
historyTypeId	The ID of the Business Object type that holds the history.
queueStandInKey	The internal key for a queue derived from its scope, scope owner, and folder.

Quick search operations (operation configuration, requests, or responses)

Parameter	Operation Usage
allowQuickSearch	A flag to indicate if quick search is allowed.
allowSpecificSearch	A flag to indicate if specific search is allowed.
ascending	A flag to indicate results are sorted in ascending order.
changedLimits	The limits based on changes made in a specific time frame.
columns	A list of properties for each field on the Business Object grid. Columns are returned when the schema flag it set to true.
defaultToQuickSearch	A flag to indicate that if both quick search and specific search are allowed, quick search is the default (true).
docRespositoryItemID	The internal ID for a document repository.
groups	A set of objects and a results list for a simple text search.
hasAnyOptions	True if search configuration option is set to Display (2) or UseAndDisplay (3). Option Key • 0 = None (Not selected and cannot select.) • 1 = Use (Selected and cannot clear.) • 2 = Display (Not selected and can select.) • 3 = UseAndDisplay (Selected and can clear.)
IncludeAvailableInSpecific	A flag to indicate whether available quick search items are included.
IncludeQuickSearchInSpecific	A flag to include all items in quick search in specific searches.
isBusObTarget	A flag to indicate that the quick search is based on a Business Object (true) or a document repository (false).
key	The name of a field used for sorting.
nonFinalStateOption	A flag to indicate if closed records are excluded. True excludes closed records. Use the Option Key to determine if you can change this option.
quickSearchId	The internal ID of a quick search configuration.
quickSearchItems	A set of object for each quick search configuration.
quickSearchWatermark	The text provided for the quick search control.
resolvedQuickSearchWatermark	The text provided for the quick search control. Custom text is shown if it was provided.
rowColor	The row color assigned to a record returned by the search.
rows	A list or records returned based on search criteria.

Parameter	Operation Usage
searchAnyWordsOption	A flag to indicate whether any or all words will be used in a query. Use the Option Key to determine if you can change this option.
searchAttachmentsOption	A flag to indicate whether attachments will be queried. Use the Option Key to determine if you can change this option.
searchRelatedOptions	A flag to indicate whether related Business Objects will be queried. Use the Option Key to determine if you can change this option.
searchResultsFieldValues	A list of field value objects returned for a specific search. Objects are fieldId, name, value, and the dirty flag.
searchTargetId	The Business Object ID or document repository to query.
searchTargetType	Either Business Object or document repository.
selectedChangeLimit	The change time frame for the query. For example: `{ "displayName": "Yesterday", "units": "Days", "value": -1 },` Possible values are based on the unit. In the example above, -1 for the Days unit equals today minus 1 day.
selectedSortByFieldId	The fieldID of a field used to sort results.
simpleResultsListItems	A list of records returned for a query.
sortByFields	A list of pre-defined options available for the search.
sortByOption	An option to indicate if sorting is allowed. Use the Option Key to determine if you can change this option.
specificSearchItems	A list of options and settings for specific Business Object quick searches.
subtitle	The Last Modified Subtitle for a search result record.
targetId	The Business Object ID to query.
title	The search criteria for a simple text search.
useSortBy	A flag to indicate if sorting should be used (true) or not (false).
watermarkText	The text provided for the quick search control for specific searches.

Search operations (search item or search result operation responses)

Parameter	Operation Usage
allowValuesOnly	A flag to indicate that only specified values can be used for a prompt (true) or that any value can be provided (false).
changedOption	An indicator that a change limit can be used. Use the Option Key to determine if you can change this option. Option Key • 0 = None (Not selected and cannot select.) • 1 = Use (Selected and cannot clear.) • 2 = Display (Not selected and can select.) • 3 = UseAndDisplay (Selected and can clear.)
childFolders	A list of child folders that contain stored queries.
childItems	A list of stored queries that are children of a scope.
constraintXml	An XML version of properties for a constraint used for a prompt.
customGridDefId	The internal ID for the grid used to override field list settings.
default	The default value set for a prompt.
defaultSortOrderAscending	A flag to indicate the default sort order. True is ascending; false is descending.
exportFormat	The format of exported searches: • 0 = CSV • 1 = Excel • 2 = Tab • 3 = Word • 4 = Custom Separator • 5 = Simple JSON
fields	A list of fields to return.
filters	A set of filters to apply to searches. Each filter includes fieldId, operator, and value.
hasPrompts	A flag that indicates that a stored query has prompts.
includeAllFields	A flag to include all fields in impromptu search results.
includeSchema	A flag to include the table schema of the saved search. If false, results contain the fieldid and field value without field information. Default is false.
isDateRange	A flag to indicate that a prompt includes a date range.

Parameter	Operation Usage
listDisplayOption	The list display option for a prompt: • 0 = Auto • 1 = Simple text box • 2 = List of values in a combo • 3 = List of values in a grid • 4 = List of values in a simple list
listReturnFieldId	The internal ID for the field returned when a prompt list is a grid.
localizedScopeName	A translated scope name based on a user's assigned culture.
parentFolderId	The name or internal ID of a folder that contains stored queries.
promptId	The internal ID or name for a prompt.
promptName	The display name of a prompt.
prompts	A list of objects associated with prompts.
promptType	The type of prompt: • 1 = Text • 2 = Number • 3 = Date/Time • 4 = Logical • 5 = Date • 6 = Time
promptValue	The default value for a prompt.
promptValues	For search operation requests, a prompt ID, prompt name, and value. Either prompt ID or prompt name is required. This can be a list of multiple prompts needed for each request. For search operation responses, a list of values to choose from. Values are not processed at runtime.
searchID	The internal ID of a stored query.
searchName	The display name of a stored query.
searchResultsFields	A list of field schema objects.
searchTerm/searchText	A text string used to filter search results.

Parameter	Operation Usage
sortDirection	The possible values are: • 0 = No sorting • 1 = Ascending • 2 = Descending
sorting	A set of objects used to sort search results.
supportedAssociations	A list of Business Objects associated with the Business Object you are querying.
text	The text shown in the prompt window.
totalRows	The number of records returned by a search.

Security operations (requests or responses)

Parameter	Operation Usage
accountLocked	A flag to indicate that a user account is locked.
add	A flag to indicate if "Add" is granted to a security group for a specific right.
addUserToTeamRequests	A list of objects for adding users to teams.
allow	A flag to indicate if "Allow" is granted to a security group for a specific right.
applicationtype	The type of application to get authentication settings for. Values are RichClient (CSM Desktop Client, BrowserClient, BrowserPortal (CSM Portal), and MobileClient (Cherwell Mobile).
browserClientCustomViewId	The internal ID of the Browser Client View assigned to a role.
businessObjectExcludeList	The internal IDs for Business Objects that are excluded for a role.
categoryid	The internal ID for the security group category.
categoryname	The display name for the security group category.
createDateTime	The date and time a user account was created.
culture	The language code for the culture assigned to a role.
delete	A flag to indicate if "Delete" is granted to a security group for a specific right.
departmentMemberEdit	A flag to indicate if a department member can edit Business Object records or fields in those records.
departmentMemberView	A flag to indicate if a department member can view Business Object records or fields in those records.

Parameter	Operation Usage
edit	A flag to indicate if edit rights are granted.
emailAlias	The email alias assigned to a team
fieldPermissions	A list of rights objects for a security group.
groupid	The internal ID for a security group.
groupname	The display name for a security group.
image	The image assigned to a team.
internalLoginAllowed	A flag to indicate if internal authentication is configured for a specified client.
isYesNoRight	A flag to indicate that a security group right is either allowed or not allowed.
lastPasswordResetDate	The date a user's password was last reset.
lastPasswordResetTime	The time a user's password was last reset.
ldapLoginAllowed	A flag to indicate if LDAP authentication is configured for a specified client.
dlapRequired	A flag to indicate whether Active Directory group membership is required.
managerOfOwnerEdit	A flag to indicate if a manager of an owner can edit Business Object records or fields in those records.
managerOfOwnerView	A flag to indicate if a manager of an owner can view Business Object records or fields in those records.
nextPasswordResetDate	The date on which users are required to change their passwords.
nonScopeOwnerAdd	A flag to indicate that users are granted the Add right for items outside of their ownership scope.
nonScopeOwnerDelete	A flag to indicate that users are granted the Delete right for items outside of their ownership scope.
nonScopeOwnerEdit	A flag to indicate that users are granted the Edit right for items outside of their ownership scope.
nonScopeOwnerView	A flag to indicate that users are granted the View right for items outside of their ownership scope.
ownerEdit	A flag to indicate if an owner can edit Business Object records or fields in those records.
ownerView	A flag to indicate if an owner can view Business Object records or fields in those records.
passwordNeverExpires	A flag to indicate whether a user's password expires or not.
primaryBusObId	The internal ID of the primary Business Object assigned to a role.

Parameter	Operation Usage
publicid/publicId	The public ID used to identify users.
rightId	The internal ID for a security group right.
rightName	The internal name for a right.
roleId	The internal ID for a role.
roleName	The display name for a role.
roles	A list of objects related to roles.
samlLoginAllowed	A flag to indicate if SAML authentication is configured for a specified client.
SecurityGroupId	The internal ID for a security group.
securityGroups	A list of objects related to security groups.
shortDisplayName	The display name for a user.
smartClientCustomViewId	The internal ID of the Desktop Client view assigned to a role.
standardRightName	The internal name for a right, prepended with the right's category.
teamEdit	A flag to indicate if a team member can edit Business Object records or fields in those records.
teamId	The internal ID for a team.
teamManagerOfOwnerEdit	A flag to indicate if a team manager can edit field in a Business Object record.
teamManagerOfOwnerView	A flag to indicate if a team manager can view field in a Business Object record.
teamName	The display name for a team.
teams	A list of objects related to teams.
teamType	The type of team: • User = User teams • CustomerWorkgroup = Customer workgroups
teamView	A flag to indicate if a team member can view Business Object records or fields in those records.
userCannotChangePassword	A flag to prevent users from changing their passwords.
userInfoFields	A list of objects related to user information fields.
userIsTeamManager	A flag to indicate that a user is a manager of a team.
userMustChangePasswordAtNextLogin	A flag to require users to change their password when they log in next.
view	A flag to indicate if view rights are granted.

Parameter	Operation Usage
viewRunOpen	A flag to indicate if Run, View, or Open are granted to a security group for a specific right.
windowsLoginAllowed	A flag to indicate if Windows authentication is configured for a specified client.
windowsUserId	The Windows login ID for a CSM user.

Service information operations (responses)

Parameter	Operation Usage
apiVersion	The version of the Cherwell REST API you are using.
csmCulture	The default culture for CSM.
csmVersion	The CSM version.
systemDateTime	The current date and time for the CSM server in UTC format.
systemUtcOffset	The UTC offset so the system can transform a date/time.
timeZone	The time zone of the CSM Application Server. Data returned is a serialization of the .NET TimeZoneInfo class.

Token operations (requests or responses)

Parameter	Operation Usage
.expires	The date and time a token expires.
.issued	The date and time a token was issued.
access_token	The issued access token.
as:client_id	The API Client ID used to request a token.
auth_mode	The authentication mode to use for requesting access tokens. Options are Internal (for CSM authentication), Windows, LDAP, and SAML.
client_id	The API client ID for the client making the token request.
client_secret	The API client secret for the native client making the token request. This is only required for native clients.
expires_in	The number of seconds before a token expires.
grant_type	The type of token being requested: password or refresh token.
refresh_token	The refresh token used to grant another access token.
token_type	The type of token. Currently, the type is always "bearer".

Type

Parameter	Operation Usage
type (Attachments)	The type of record attachment: • 0 = None: Not applicable to the Cherwell REST API. • 1 = File: Linked files. • 2 = FileManagerFile: Imported files. • 3 = BusOb: Attached Business Objects. • 4 = History: Information about the attachment, if available. • 5 = Other: Not applicable to the Cherwell REST API.
type (Business Objects)	The type of Business Object: • All • Major • Supporting • Lookup • Groups
type (Fields)/fieldType	The type of field (Text, Date/Time, etc.).

Miscellaneous

Parameter	Operation Usage
attributes	A list of attributes defined for specific fields.
busobname/busObName	The display name for a Business Object type. Examples include: • Incident • Task • Service Cart
categoryDescription - getsecuritygroupcategories	Not used by the Cherwell REST API.
dateTimeFormatting	For ExportSearchResults only, the format for Date/Time fields. For example: mm/dd/yyyy or dd/mm/yyyy.  **Note:** Due to other system changes, this parameter has been deprecated.
deleteRequests	A list of deleteRequest objects that will be processed.

Parameter	Operation Usage
description	The description of an object.
displaytext/displayName	The display text for the item. For example, use the attachment name, which is the display text for an attachment record.
error	An error message. (Deprecated in V2 operations.)
errorCode	A text string used to indicate a type of error message.
errorMessage	An error message.
fieldName	The display name of a field.
fields	In most cases, set a field parameter, such as *fieldId*, *name*, and *value*.
fieldsList	A specific list of fields to request.
fieldValidationErrors	A list of objects related to field validation.
fullFieldiD	A string containing the internal Business Object ID and internal Field ID.
height	For Get Gallery Image operation requests, the height of an icon.
lang	Returns localized responses in the specified language.  **Note:** Language codes must be passed using the en-US format (fr-FR, de-De, etc.). Language codes are not case-sensitive.
locale	Returns localized responses (date/time, currency, number formatting) in the specified locale.  **Note:** Language codes must be passed using the en-US format (fr-FR, de-De, etc.). Language codes are not case-sensitive.
loginId	A user's login ID.
mobileClientCustomViewId	Not applicable to the Cherwell REST API.
operator	A string used to evaluate filters.
pageNumber	The page number of the result set to return.
pageSize	The number of rows to return per page.
parentBusObPublicId	The public ID for the parent Business Object.
parentbusobrecid	The record ID for the parent Business Object.
password	The password assigned to a user account.
readRequests	A list of readRequest objects used to return a batch of items, such as Business Objects or users.

Parameter	Operation Usage
recIdFields	The record ID fields specified for a Business Object.
relationshipid	The internal Relationship ID for the related Business Object.
scanCode	The scan code for a specific Business Object record.
standInKey/ oneStepActionStandInKey	For Core operations, the internal key for an item derived from its scope, scope owner, and folder.
totalRecords	For batch delete operations, the total number of records deleted.
url	A URL string.
username	A CSM user's Login ID.
value	A string used to identify a value or to filter results.
width	For Get Gallery Image operation requests, the width of an icon.

Cherwell REST API Performance

Examples of poor performance might be API calls, such as the first call made to CSM, taking longer than expected to complete. This can occur if an application pool times out while simultaneously an API operation is trying to execute. This scenario causes the pool to restart, which can cause a delay.

You can resolve this issue by completing the following steps. If these steps do not resolve the issue, you can alternatively create scheduled items that execute every 20-30 minutes to execute an API call. You could call the ServiceInfo operation, for example.

To ensure optimal Cherwell REST API performance:

1. On the server where the Cherwell REST API is installed, open the **Internet Information Services (IIS) Manager** (under **Administrative Tools**).
2. Expand the branch for the relevant server and select **Application Pools**.

   The Application Pools**CherwellAPI** page is displayed.
3. Right-click and select **Advanced Settings**.

   The **Advanced Settings** window is displayed.
4. Replace the default **Idle Time-out (minutes)** setting. We recommend `100`.

   If more time is required to complete a large operation, enter the required amount of time in minutes.

    **Note:** The default Microsoft setting is `20` minutes.
5. Set **Start Mode** to `AlwaysRunning`.
6. Set the **Regular Time Interval (minutes)** to `0`.
7. Select **OK**.

   The **Advanced Settings** window closes, and you are returned to the **Internet Information Services (IIS) Manager** showing the Application Pools page.
8. To implement the changes, right-click **CherwellAPI** and select **Recycle**.
9. Setting an automatic recycle schedule is important because running it indefinitely causes the pool to consume more resources than necessary. Accordingly, to set up when the automatic recycle will occur, right-click **CherwellAPI** and select **Recycling**.

   The **Edit Application Pool Recycling Settings** window is displayed.
10. Select *only* the **Specific time(s)** check box (clear all other check boxes), and then enter a specific time. We recommend `2:00 AM`. Another example is `11:00 PM`.
11. Select **Next > Finish**.