Oracle Life Sciences Empirica

Topics REST API Guide

Release 2025.4.01
G39969-01

ORACLE

Oracle Life Sciences Empirica Topics REST API Guide, Release 2025.4.01

G39969-01

Contents

Preface

This preface contains the following sections:

Documentation accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Related resources

All documentation and other supporting materials are available on the Oracle Help Center.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through Oracle Support Cloud.

Contact our Oracle Customer Support Services team by logging requests in one of the following locations:

- English interface Customer Support Portal (https://hsgbu.custhelp.com/)
- Japanese interface Customer Support Portal (https://hsgbu-jp.custhelp.com/)

You can also call our 24x7 help desk. For information, visit https://www.oracle.com/life-sciences/support/ or visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab if you are hearing impaired.

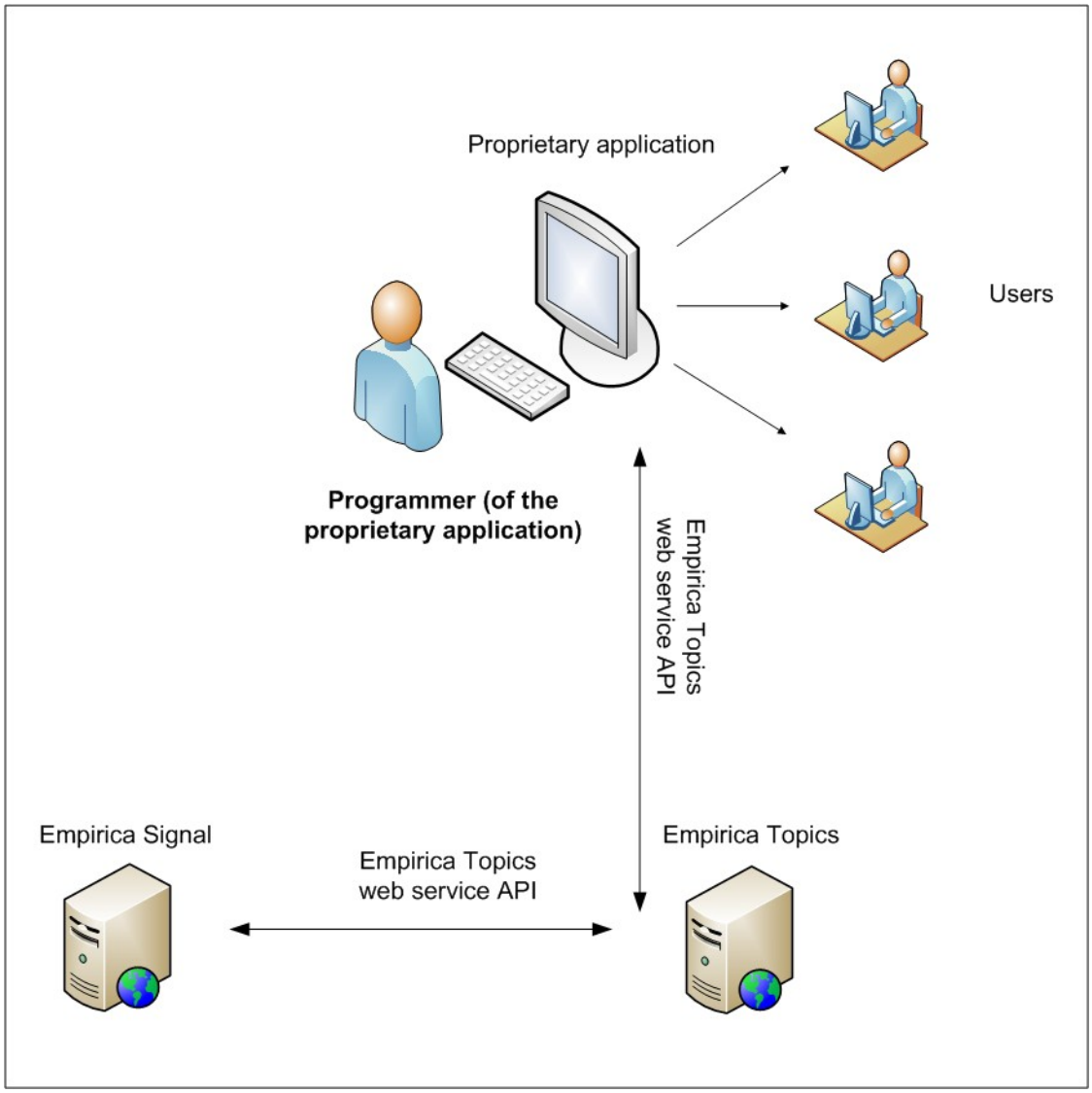
1

Web service configuration

Overview

Using the Oracle Empirica Topics web service, you can integrate your proprietary application with Oracle Empirica Topics for users to attach artifacts from your application to new or existing topics or actions.

Figure 1–1 API overview



In your application, you must add integration points that implement the Oracle Empirica Topics REST API.

In this document, code snippets are displayed for Java. For a .NET application, C# techniques are comparable. Calling the API via Python or PL/SQL is also supported.

> **Note**
>
> Oracle Empirica Topics is installed as part of the Oracle Empirica Signal application and includes:
>
> - The Empirica Topics web service that implements the Oracle Empirica Topics REST API for Save to Topic in your proprietary application.
> - Topic Management in Oracle Empirica Signal for end users to view and edit their topics.
> - Topics administration in Oracle Empirica Signal for administrators to manage topic workflow configurations, work teams, user accounts, and user permissions.

Configuring users

The Oracle Empirica Topics web service acts on behalf of your application's end users. Any of your application's end users that are accessing topics through your application must also be provisioned with the same username and the appropriate Oracle Empirica Topics permissions and work teams on the Oracle Empirica Signal server.

The Oracle Empirica Topics web service validates these Oracle Empirica Topics-specific privileges and work team permissions for that username. When you implement an API call, you must set the **Username** field in the `TopicsServiceContext`. For most tasks, you must also set the **twcId** field in the `TopicsServiceContext`. For more information, see API. If your application tries to access a topic on behalf of either an unknown username or a user without valid permissions, a `TopicsServiceException` is generated. Only topics, actions, topic templates, and work teams that the user has permission to access are returned from any call.

Security

The Empirica Topics REST API supports two authentication methods, Basic and oAUTH 2.0. For cloud installations, the service is configured for use with oAUTH only using IDCS as the token provider.

Basic Authentication

The WebLogic administrator must create a user for exclusive use by your application and the Oracle Empirica Topics web service as described in the *Empirica Installation and Upgrade Instructions*.

The Oracle Empirica Topics web service username and password credentials are provisioned on the Oracle Empirica Topics WebLogic server. Your application then sets this username and password, along with the standard WS-SECURITY Username Token Policy, as part of every web service call. The Oracle Empirica Topics server validates access to the Topics web service using these credentials. The Oracle Empirica Topics server can return an exception if the web service credentials are not correct or the Username Token Policy has not been set. The Oracle Empirica Topics web service will then not be available to your application.

oAUTH 2.0 Authentication

The Oracle Empirica Signal TopicsService application must be configured to specify the oAUTH authentication provider ,as described in the *Empirica Installation and Upgrade Instructions*, with the necessary URLS. Your application generates an oAUTH token from the token provider. The ClientID and ClientSecret for the token provider must be known to your application. Your application calls Empirica Topics API endpoints using the generated token. The token is then validated by the Empirica Topics Server. The Oracle Empirica Topics server returns an exception if the token is not a valid one.

Connecting to the Oracle Empirica Topics web service

To connect your application to the Oracle Empirica Topics web service, you need to set up web service properties in your code.

- **The end point address**—This is the URL address of the Oracle Empirica Topics web service such as
  `https://<hostname >:<port number /TopicsServiceR/empirica`

It is recommended that the Oracle Empirica Topics web service run with SSL so that your application attaches to the service using HTTPS.

- Authentication Header Credentials
  Basic Authentication:
  - **Username**—This is the Oracle Empirica Topics service user name.
  - **Password**—This is the Oracle Empirica Topics service password.
- **oAUTH Authentication:**
  - **Authentication Provider URL**
  - **Client ID**
  - **Client Secret**
  - **Scope**
- Java integration
  The following example demonstrates how to connect to the Oracle Empirica Topics web service using Jakarta as the JAX-RS implementation provider.

Java integration

The following example demonstrates how to connect to the Oracle Empirica Topics web service using Jakarta as the JAX-RS implementation provider.

// Create new client

```
Client client = ClientBuilder.newClient();
```

// Set the endpoint address for the Topics web service

```
String target = https://<hostname>:<portnumber>/TopicsServiceR/
empirica”
```

+ additional path parameters for the specific call

// Get the authorization string for the header

```
String authorizationTokenOrString =
```

- **For Basic Auth:** `"Basic " + Base64.encodeBytes(“<topics service username >:<topics service password >”.getBytes());`
- **For oAuth:** `"Bearer "` + oAuth token obtained from your provider (see example below)

// Invoke the Topics REST web service call. In this example a POST method, using JSON Input and Output.

```
String InputJSON = Input parameters in JSON format

Response response = client.target(target)

.request(MediaType.APPLICATION_JSON_TYPE)

.header("Authorization", authorizationTokenOrString)

.post(Entity.entity(inputJSON, MediaType.APPLICATION_JSON_TYPE));

String outputJSON = response.readEntity(String.class);
```

Substitute your environment's values for <hostname >, <port number >, <topics service username >, and <topics service password >.

Your application must handle serializing the input to JSON format, and deserialization from the response.

// Check the status of the response

```
response.getStatusInfo().getStatusCode()
```

// sample code to get an oAuth access token

```
String target = "your-token-provider-url";
String clientId = "your-client-id";
String clientSecret = "your-client-secret";
String scope = "your-scope";
String access_token = "";
String form = "grant_type=client_credentials" +
      "&client_id=" + URLEncoder.encode(clientId,
         StandardCharsets.UTF_8) +
      "&client_secret=" + URLEncoder.encode(clientSecret,
         StandardCharsets.UTF_8) +
      "&scope=" + URLEncoder.encode(scope, StandardCharsets.UTF_8);

 client = ClientBuilder.newClient();
 Response tokenResponse =
    client.target(target)
    .request(MediaType.APPLICATION_JSON)
    .post(Entity.entity(form,
         MediaType.APPLICATION_FORM_URLENCODED_TYPE));
 String jsonString = tokenResponse.readEntity(String.class);
 // parse JSON to get the "access_token" attribute
```

2

Use cases

Implementing the Save to Topic features

Using Topic Management in Oracle Empirica Signal, an end user can create, display, edit, and move topics through a workflow. Topics can have attachments such as files, graphs, tables, URLs, or notes. Topics can also contain one or more actions that can also have attachments.

A topic workflow configuration contains all attributes of topics. Topics are associated with work teams, which are sets of users with permissions to view, edit, or save an attachment to a topic. An Oracle Empirica Signal administrator defines topic workflow configurations and configures work teams and their permissions.

Your proprietary application can use the Oracle Empirica Topics web service to display the **Save to Topic** link and dialog boxes for a user of your application to attach tables, graphs, or files to topics and/ actions.

The use cases in this chapter describe a sample implementation by Oracle Empirica Signal of the Save to Topic features and associated web services calls. You can integrate with Save to Topics in your user interface in a way that is appropriate for your proprietary application.

Save to Topic link

Your application can incorporate a **Save to Topic** link into its pages, similar to how Oracle Empirica Signal includes the link above the Data Mining Results Table.

If the user clicks the **Save to Topic** link, the application displays a dialog box and calls the Oracle Empirica Topics web service. The Oracle Empirica Topics web service initializes the

values and saves an attachment containing the application's data to the specified topic or action.

Tasks

To implement the **Save to Topic** link, your application invokes the `getUserInfo` task.

Your application should display the **Save to Topic** link conditionally, based on the user's permissions. The link should appear only if the user has the appropriate work team permissions.

```
TopicsUserInfo.isCanSavetoTopic = true
```

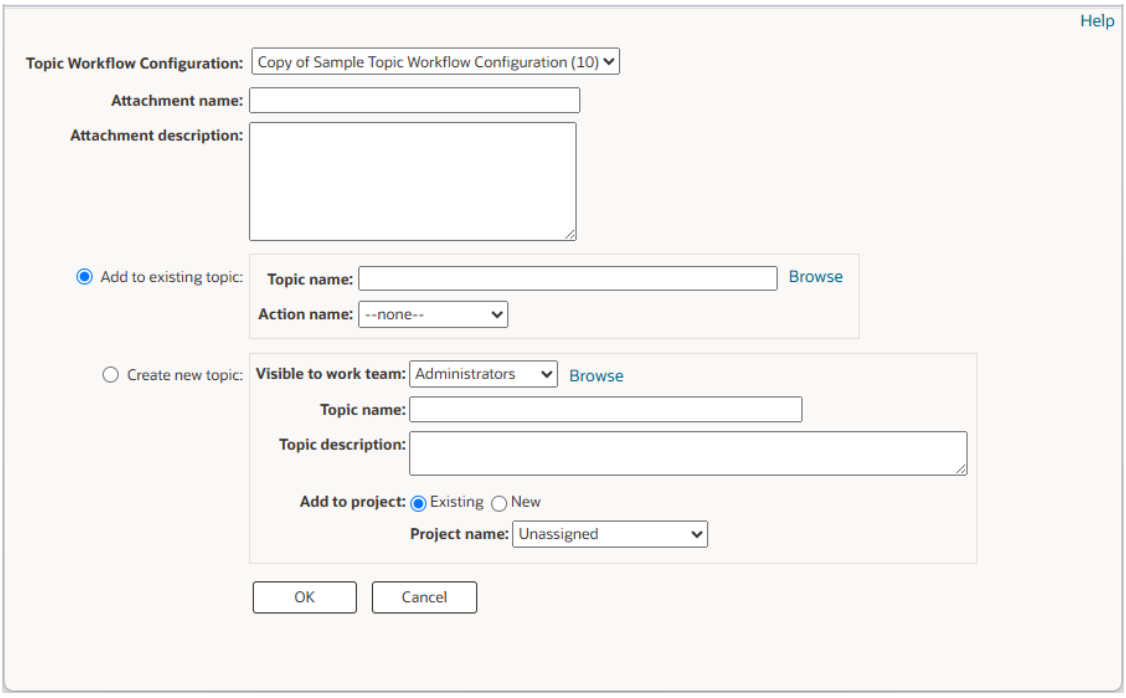
The user must have the same username in your application and in Oracle Empirica Topics. `getUserInfo` gives a `TopicsServiceException` if the username from your application is not found.

Save to Topic dialog box

When the user selects **Save to Topic**, the Save to Topic dialog box appears. Here, the user can save to an existing topic or action, or to a new topic.

Your application should create a Save to Topic dialog box. This page should have similar options to the example (Figure 2–1) and can vary based on your application's design.

Figure 2-1 Save to Topic dialog box



To save to a topic or action, the user can enter the attachment name and select an existing topic from the **Add to existing topic** drop-down list, or create a new topic. If an existing topic is selected, the **Action name** drop-down list is populated with the actions for that topic. The user can save an attachment to a topic or to an action in the topic.

If the user selects the **Create new topic** option, a new topic name is required and the user either selects an existing project or a new project. Projects are used to categorize a topic and are optional. The user can select **Unassigned** from the drop-down list if there is no project. The project, **Unassigned**, has ID 0. To create a topic, the user must have the permission to create topics.

Tasks

Your application invokes the following tasks to render the Save to Topic dialog box.

- `getUserInfo`—Gets the user privileges. The task returns the `TopicsUserInfo` value object, which contains the field `CanSaveToNewTopic`. If this field is false, you can disable the section for creating topics.
  The field `allowedAttachments` contains a list of the type of attachment types that are allowed for this user configuration. The values in the list can be TABLE, IMAGE, URL, FILE, NOTE, or SIGNAL. The SIGNAL type is reserved. Use this list to determine the types of attachments to include in your dialog. For the Save to Topic example in Figure 2-2, only TABLE and IMAGE are included.

  If `allowedAttachments` includes URL, FILE, and NOTE, then a user selects the attachment type from the **Attachment type** radio button. The user selects the radio button **Application data** for a table or image from the application, or selects **file**, **note**, or **URL**. The **file** and **URL** radio buttons have input fields to enter the file or URL. For the **note** type, the user enters the text for the note in the attachment description field.
- `getTopicConfigs`—Gets the list of TopicWorkflowConfigurations available to the user, which can be used to populate the TopicWorkflowConfiguration dropdown list.
- `getTopicContexts`—Gets all topics to which the user can save an attachment and populates the existing topics drop-down list.
  Oracle Empirica Signal applications remember the most recent topic when the user saves an attachment. This topic becomes the default topic since users often save additional attachments to the most recent topic. You may design your application to remember the most recent topic.
- `getActionContexts`—Gets associated actions using the user's selected existing topic.
  Oracle Empirica Signal applications remember the most recent saved action when the user saves an attachment. This action becomes the default action since users often save additional attachments to the most recent action. You can design your application to remember the last saved action.
- `getProjects`—Gets a list of project names associated with your topics. This task is used when creating a topic.
  Oracle Empirica Signal applications remember the most recent project when the user saves an attachment to a new topic. This project becomes the default project for new topics since users sometimes create additional topics within the most recent project.

  When Oracle Empirica Signal applications populate the **Project** drop-down list, they combine the project names returned from the Oracle Empirica Topics web service with projects from other areas of the application. In this way, users can associate new topics with projects across the application. If your application categorizes objects by name, you can design your application to aggregate project names from topics with other names from your application and thus include all these category names in the **Project** drop-down list. If you specify a project name that does not already exist, the project name is saved as a new project when the topic attachment is saved.
  You can design your application to remember the most recent project when new topics are created.
- `getTopicTemplateContexts`—Gets the topic templates visible to at least one work team to which the user belongs.

If topic templates have been created by an Oracle Empirica Topics administrator, and they are visible to a work team to which the user belongs, then the **Topic template** drop-down list should appear in the Save to Topic dialog box.

In the Save to Topic dialog box, if the user selects **OK**, the attachment is saved to the specified topic or action.

Your application invokes the following tasks to save the attachment when the user selects **OK** in the Save to Topic dialog box.

- `attachTopic`—Attaches to an existing topic.
- `attachAction`—Attaches to an existing action.
- `attachNewTopic`—Creates a new topic and adds the attachment to it.
- `attachNewAction`—Creates a new action and adds the attachment to it.
- Streaming attachments
  To save IMAGE or FILE attachments, your application must first call the Oracle Empirica Topics service API `streamFileAttachment`, and then call one of the four Oracle Empirica Topics web service calls to attach the file to a topic or an action.
- Save to topic and work teams
  An Oracle Empirica Topics administrator can configure a topic workflow configuration such that the topics are visible to a single work team, more work teams, or no work teams.

Streaming attachments

To save IMAGE or FILE attachments, your application must first call the Oracle Empirica Topics service API `streamFileAttachment`, and then call one of the four Oracle Empirica Topics web service calls to attach the file to a topic or an action.

The second call must be invoked within the streamingTimeOut time limit, as configured for your service, which defaults to three seconds.

Save to topic and work teams

An Oracle Empirica Topics administrator can configure a topic workflow configuration such that the topics are visible to a single work team, more work teams, or no work teams.

If a topic workflow configuration is set to allow a topic to be visible to **one and only one work team**, the user must select a work team when creating a new topic from within your application. A **Visible to work team** drop-down list appears in the Save to Topic dialog box when creating new topics.

If a topic workflow configuration is set to allow a topic to be visible to **Zero, one, or more work teams**, then no **Visible to work team** drop-down list should appear. A new topic is private to the user who created it or to a superuser. Either of these users can edit the topic from within the Oracle Empirica Signal Topics application and publish it by assigning it to one or more work teams.

Select Topic dialog box

From the Save to Topic dialog box, the user can browse for existing topics. From the **Browse** link, the Select Topic dialog box appears where the user can select a topic or can view all properties for available topics.

The Select Topic dialog box from Oracle Empirica Signal can include filter fields to limit the topics that appear in the Select Topic table. When the user selects a topic and selects **OK**, the topic appears in the **Topic name** field of the Save to Topic dialog box.

Tasks

Your application invokes the following tasks to render the Select Topic dialog box.

- `getFieldMetadata`—Gets the topic field definitions that are used to populate the column headers in the Select Topic table.
- `getFieldValues`—Gets the topic field values to populate the rows in the Select Topic table.
- `getFilters`—Gets topic field definitions and values for topic fields that can be used as filters. An administrator can designate topic fields as filters. These fields can be displayed in this dialog so that a user can limit the list of topics based on the filters.

> **Note**
>
> For the above calls, set the `FieldContext` parameter for the web service call to `FieldContext.TOPICS`.

Browse work teams

In the Save to Topic dialog box, users can browse work teams.

The user selects **Browse** to navigate to the Browse Work Teams dialog box and select a work team. When the user selects a work team and selects **OK**, the work team appears in the **Visible to work team** drop-down list.

Tasks

Your application invokes the following tasks to render the Browse Work Teams dialog box.

- `getFieldMetadata`—Gets the work team field definitions that are used to populate the work team column headers in the browse work team table.
- `getFieldValues`—Gets the work team field values that are used to populate the work team rows in the browse work team table.
- `getFilters`—Gets work team field definitions and values for work team fields that can be filtered. An administrator can designate work team fields as filters. These fields are displayed in this dialog so that a user can limit the list of work teams based on the filters.

> **Note**
>
> For all the above calls, set the `FieldContext` parameter of the task to `FieldContext.WORKTEAMS`.

Viewing topics or actions

Oracle Empirica Signal allows users to display a list of topics or actions.

Tasks

Your application invokes the following tasks to present a table of topics or actions.

- `getFieldMetadata`—Gets the topic or action field definitions that are used to populate the column headers in the table.
- `getFieldValues`—Gets the topic or action field values to populate the rows in the table.

3
API

API summary

The API includes topic service tasks that retrieve information about topics or actions, or save to a topic or action.

All input is validated, including checks to verify that a user has permission to save to topic or action. Any data that is retrieved is based on the user's work team permissions and topic workflow configuration permissions.

Table 3-1 summarizes the Topics service tasks.

> **Note**
>
> All endpoints are POST tasks, except for `getProperties`, which is a GET.

Operation	Task Name	Description	Path (BaseURL +)	Input	Response
Get a list of topics	getTopicContexts	Retrieves topic names and IDs.	/topics @QueryParam: findTopicId	TopicsServiceContext	TopicContexts
Get a list of actions	getActionContexts	Retrieves action names and IDs for a topic, for which user can add an attachment.	/actions @QueryParam: topicId	TopicsServiceContext	ActionContexts
Get a list of work teams	getWorkteamContexts	Retrieves work team names and IDs.	/workteams	TopicsServiceContext	WorkteamContexts
Get a list of topic templates	getTopicTemplateContexts	Retrieves topic template names and IDs.	/topictemplates	TopicsServiceContext	TopicTemplateContexts

Operation	Task Name	Description	Path (BaseURL +)	Input	Response
Attach to an existing topic	attachTopic	Creates an attachment to an existing topic.	/topic/attach	{TopicsService Context	Empty (status only)
Attach to a new topic	attachNewTopic	Creates an attachment to a new topic. Creates a topic without an attachment.	/topic/attachnew	{TopicsService Context, TopicInput, AttachmentInput}	TopicResult
Attach to an action	attachAction	Creates an attachment to an existing action.	/action/attach	{TopicsService Context, ActionContext, AttachmentInput}	Empty (status only)
Attach to a new action	attachNewAction	Create the action, and optional attachment, added to existing topic.	/action/ attachnew	/file @QueryParam: guid	Empty
Stream a file	streamFileAttachment	Create a temporary file.	/file @QueryParam: guid	BINARY_INPUT _STREAM	Empty
Get field metadata	getFieldMetadata	Retrieves field properties for topics, topic templates, or work teams.	/{context}/field Where context is a Field Context value Where context is a Field Context value {topics, topicTemplates, workteams, actions}	TopicsServiceContext	FieldMetadata
Get field values	getFieldValues	Retrieves field values for topics, topic templates, or work teams.	/{context}/value Where context is a Field Context value {topics, topicTemplates, workteams, actions}	TopicsServiceContext	FieldValues
Get filters	getFilters	Retrieves fields defined as filters and their values for topics, topic templates, workteams or actions.	/{context}/filter Where context is a Field Context value {topics, TopicTemplates, Workteams, actions}	TopicsServiceContext	Filters
Get a list of projects	getProjects	Retrieves projects used by the user's topics.	/project	TopicsServiceContext	ProjectList

Operation	Task Name	Description	Path (BaseURL +)	Input	Response
Get a list of action types	getActionTypes	Retrieves Active Action Types visible to the user.	/actionTypes	TopicsServiceContext	ActionTypeContexts
Get user info	getUserInfo	Retrieves topic permissions for the user; for example, can the user save to a topic.	/user	TopicsServiceContext	TopicsUserInfo
Get topics service properties	getTopicsServiceProperties	Retrieves the Oracle Empirica Topics service version and other service properties.	/props	[None]	TopicsServiceProperties
Get a list of topic workflow configurations (TWCs)	getTopicConfigs	Retrieves a list of TWCs available to the user.	/twc	TopicsServiceContext	TopicConfigContexts

Common value objects

Oracle Empirica Topics service tasks contain value objects that are passed as arguments or returned from service calls. This section describes the member variables for each value object.

- TopicsServiceContext value object
  Most tasks have a `TopicsServiceContext` value object passed as a parameter.
- TopicsSortOrder value object
  An end user can sort the topic, topic template, work team, or action field values. Your application sets the **TopicsServiceContext.sortOrder** field with a list of fields to sort and whether to sort each field in ascending or descending order. If the order is not specified, the default is ascending order.
- TopicsFilter value object
  An end user might specify a filter, for example, when looking for topics with a drug name in a custom field.
- AttachmentInput value object

TopicsServiceContext value object

Most tasks have a `TopicsServiceContext` value object passed as a parameter.

This value object contains context information to process a request. The **username** field is required. For any task that is specific to a topic workflow configuration, the **twcId** field is required. You can also set optional parameters.

Table 3–1 TopicsServiceContext optional parameters

Field name	Parameter type	Description
applicationName	Input	Name of your proprietary application. This field is appended to the source metadata for an attachment. For example, if the **AttachmentInput.source** field is specified as `Results`, the source is modified to be `Results - My Application Name`. When viewing attachments in Oracle Empirica Topics, this field enables a user to distinguish attachments from a variety of applications.
applicationInstance	Input	Reserved for future use.
version	Input	Version of the Oracle Empirica Topics API that you are using. Oracle Empirica Signal Release 2025.4.01 contains version 5 of the Oracle Empirica Topics API, as described in this document. If your Oracle Empirica Topics API version is greater than the Oracle Empirica Topics server version, a `TopicsServiceException` is thrown.
sortOrder	Input	List of columns on which to sort data and the sort order for each column. This field applies to only the `getFieldValues` task and is ignored for any other tasks. If this field is not specified, a default sort order is used. `fieldType` is one of: STRING, DATE, INTEGER, LONG, FLOAT, PROJECT, STATE, ACTION_TYPE `order` is one of: SORT_ASC, SORT_DESC
filter	Input	List of columns on which to filter data and the filter value to match on. This field applies to only the `getFieldValues` task and is ignored for any other tasks. If this field is not specified, the data is returned without any filtering. Numbers and dates should be converted to strings. Dates use mm/dd/yyyy format.
startRow	Input	Start row from which to begin retrieving data. If this field is not specified, the default value is 1.

Field name	Parameter type	Description
chunkSize	Input	Maximum number of rows returned by the service on any applicable calls. Use this field to page large amounts of data rather than returning everything in one call. If this field is not specified, the default value is 500 rows. The fields **chunkSize** and **startRow** apply to the `getTopicContexts`, `getActionContexts` , `getFieldValues`, `getWorkteamContexts`, and `getProjects` tasks. These fields are ignored for other tasks.
nameContainsFilter	Input	Case-insensitive wildcard filter for topic names. This field applies only to the `getTopicContexts` task and is ignored for any other tasks. If specified, this field filters the results to topic names that contain this text.
version	Output	Version of the Oracle Empirica Topics server.
maxRows	Output	Maximum number of rows for a table attachment when attaching a TABLE attachment type. This value is determined by a site option, **Max number of rows per table allowed in topic attachments**, defined on the Oracle Empirica Topics server. You can also get this property from the `getTopicsServiceProperties` task.
numRows	Output	Number of rows that were retrieved by the task. This field is less than or equal to the chunk size.
totalRows	Output	Total number of rows available for the task. You can use this value to display the total number of pages of the result based on the chunk size. This field applies to the `getTopicContexts`, `getActionContexts`, `getTopicTemplateContexts`, `getWorkteamContexts`, `getFieldMetadata`, and `getFieldValues` tasks.

TopicServiceContext JSON :

```
{
  "applicationName": "string",
  "applicationInstance": "string",
  "version": "integer",
  "username": "string",
  “twcId”: “number”,
```

```
  "maxRows": "integer",
  "startRow": "integer",
  "chunkSize": "integer",
  "numRows": "integer",
  "totalRows": "integer",
  "sortOrder": [
    {
       TopicsSortOrder object
    }
  ],
  "filter": [
    {
       TopicsFilter object
    }
  ],
  "nameContainsFilter": "string"
}
```

TopicsSortOrder value object

An end user can sort the topic, topic template, work team, or action field values. Your application sets the **TopicsServiceContext.sortOrder** field with a list of fields to sort and whether to sort each field in ascending or descending order. If the order is not specified, the default is ascending order.

Set the **fieldName** to the name of the topic, topic template, work team or action field to sort on.

Set the **sorting order** field to `SORT_ ASC` or `SORT_DESC`.

Set the **fieldType** to `STRING` for a case insensitive sort of string values. This object is used by the `getFieldValues` task. The field types are:

`STRING`, `DATE`, `INTEGER`, `LONG`, `PROJECT`, `STATE`, `ACTION_TYPE`

The field orders are: `SORT_ASC`, `SORT_DESC`

TopicsSortOrder JSON

```
{
        “fieldname” : “string”,
      “fieldType” : “string”,
      “order” : “string”
}
    ]
}
```

TopicsFilter value object

An end user might specify a filter, for example, when looking for topics with a drug name in a custom field.

After the end user selects one or more filters, set the **filter** field in the `TopicsServiceContext` when using `getTopicContexts`, `getFieldValues`, or `getProjects`.

`TopicsFilter` is a list of filter field names and the selected filter values. Set the **fieldName** to the name of the topic, topic template, work team or action field to sort on. The **fieldName** field must be one of the values in the **fieldNames** field returned from `getFilters` and the `value` must be one of the associated values returned from `getFilters`.

The field types are always returned as strings. For a type such as `ACTION_TYPE`, `PROJECT`, or `STATE`, the name is used as the string value. Date fields use “mm/dd/yyyy” format.

TopicsFilter JSON

```
{
            “fieldName” : “string”
            “value” :
            {
           “type” : “string”,
           “stringValue” : “string”
             }
```

AttachmentInput value object

Typically, your application writes an IMAGE or TABLE attachment to a temporary file, streams the file using the `streamFileAttachment` task first, and then calls the `attachTopic`, `attachNewTopic`, `attachAction`, or `attachNewAction` task.

The `attachmentInput` argument has the name, extension, attachment type, and optional properties of the attachment.

The attachment input value object includes the following fields:

- **name—**Required name field, with a maximum length of 255 characters.
- **description—**Optional description, with a maximum length of 2000 characters. If the attachment type is `NOTE`, the description field is required and contains the note text.
- **source—**Required brief name of the source, such as "Cases," with a maximum length of 2000 characters.
- **sourceText—**Required brief information about the attachment, with a maximum length of 2000 characters. This text is application-dependent and the field supports HTML tags, such as `<br>` and `<table><tr><td>` and font formatting tags. This might contain metadata to display to the user about the attachment such as the number of cases. It is displayed in the tooltip for values in the **Attachment name** column in the Topic or Action Attachments table when viewing or editing a topic.
  If the **Notes** field is not specified, then the `sourceText` is also displayed in the View Source Details dialog box. The dialog box is displayed when the user selects **View Source Details** for an attachment type.
- **type—**Required attachment type. The attachment types are:
  - **IMAGE—**A ZIP file with a series of PNG images. If a text file other than `notes*.txt` contains a comma-separated list, each item is rendered on a separate line.
  - **FILE—**Any file, such as a PDF file.
  - **TABLE—**An XML representation of a table. An example is a case series table. See XML attachment tables for the XML definition.
  - **URL—**A URL address in the form `http[s]://xxx[:port][/yyy.zzz]`, with a maximum length of 2000 characters.

  - **NOTE—**A text note, with a maximum length of 2000 characters.

- **extension—**Name of the extension, with a maximum length of 40 characters. This is required if the attachment type is `FILE`. If the type is `IMAGE`, a `ZIP` file extension is required and the zip file contains PNG images.
- **Notes—**Optional details that contain an encapsulated HTML `<table>` tag for notes. Notes are more detailed and can be much longer than the **sourceText** field since they are displayed in a scrollable window and not in a tooltip.

  If specified, the **Notes** field is displayed in the View Source Details dialog box when selecting **View Source Details** for an attachment in the Topic or Action Attachments table. If the **Notes** field is not specified, then the **sourceText** field value is displayed in the View Source Details dialog box. This field can be formatted with HTML tags, such as formatting as an HTML table and setting the font style for the text.

  The **Notes** field is different from the notes shown by selecting the **Show Notes** link above a table. The optional notes on a table are part of the XML that is streamed with the attachment. This **Notes** field is also different from the notes in IMAGE attachments, where any `notes*.txt` files in the IMAGE zip file are always displayed along with image files in the attachment.
- **guid—**Used when streaming data files. The **guid** uniquely identifies the attachment across two task calls, the first to stream the attachment and the second to attach to a topic or action.
- **urlAddress—**If the attachment type is `URL`, the **urlAddress** contains the **URL** field. It must be in the form `http[s]://xxx[port][/yyy]`. The maximum length is 2000.
- **data—**If the attachment type is `FILE`, the **data** field must be specified with a filename including the extension. This field is required for `FILE` types and is used when creating a PDF of the topic with the attachment. The attachment is displayed as a link in the PDF with the filename from the **data** field.

AttachmentInput JSON

```
{
   "name": "string",
   "description": "string",
   "source": "string",
   "sourceText":"string",
   "urlAddress": "string",
   "type": "string",
   "extension":"string",
   "notes": "string",
   "data": "string",
   "guid": "string"
}
```

Tasks

Your application invokes the following tasks to present a table of topics or actions.

- Get a list of topics
  Your application retrieves a list of topics by using the `getTopicContexts` task, which returns a list of topic contexts with the name and identifier of each topic.

- Get a list of actions
  Your application retrieves a list of actions for a topic available to the user by using the `getActionContexts` task that returns a list of action contexts with the action name and ID and the associated topic ID.
- Get a list of work teams
  Your application can retrieve a list of work teams by using the `getWorkteamContexts` task.
- Get a list of topic templates
  Your application can retrieve a list of topic templates. The topic templates can be used to prepopulate new topic fields and actions.
- Attach to an existing topic
  Your application can create an attachment to an existing topic.
- Attach to a new topic
  Your application can create an attachment to a new topic.
- Attach to an existing action
  Your application can create an attachment to an existing action by using the `attachAction` task.
- Attach to a new action
  Your application can create an attachment to an existing action by using the `attachAction` task.
- Get field metadata
  Your application can retrieve a list of all fields for topics, topic templates, work teams, or actions by using the `getFieldMetadata` task.
- Get field values
  Your application can retrieve a list of all field values for each topic, topic template, work team, or action by calling the `getFieldValues` task.
- Get filters
  Your application can retrieve a list of fields that have been set up as filters and their values by using the `getFilters` task. The Filters object is returned from the `getFilters` task and is an ordered list of filter field names.
- Get a list of projects
  Your application can retrieve a list of existing projects by using the `getProjects` task.
- Get a list of action types
  Your application can retrieve a list of action types by using the `getActionTypes` task.
- Get user info
  Your application can view the topic permissions for a user by using the `getUserInfo` task.
- Get topics service properties
  Your application can retrieve the Oracle Empirica Topics server version and other properties by using the `getTopicsServiceProperties` task.
- Get topic workflow configurations
  Your application can retrieve a list of topic workflow configurations by using the `getTopicWorkflowConfigurations` task.
- Stream file attachment
  For an attachment of type FILE or IMAGE, clients must first stream files to the server and include a GUID in the `attachmentInput` parameter, and then call one of the attach tasks.

Get a list of topics

Your application retrieves a list of topics by using the `getTopicContexts` task, which returns a list of topic contexts with the name and identifier of each topic.

This task can be used to populate a drop-down or selection list of existing topics in the **Save to Topic** dialog box. The result is alphabetically sorted by name and includes the number of values returned and the total number of values. Optionally, you can set a `start row` and a `chunk size` in the `TopicsServiceContext` value object.

This task returns all topics available to this user. To also search for a particular topic by ID, set the `findTopicId` parameter to an ID that is greater than or equal to 0.

To limit the results by name, set the **TopicsServiceContext.nameContainsFilter** field to a string. In this case, the task returns only `TopicContexts` with topic names that contain that string with a case-insensitive search. For example, when you set the **nameContainsFilter** field to `imp`, the resulting topic contexts might have results `Important Results` and Topics `imported`.

`TopicContexts` extends the value object `TopicsServiceContext` and has the actual number of rows returned in **numRows** field and the total number of rows in the **totalRows** field.

If the `findTopicId` parameter was set, then the **matchingTopicContext** field in `TopicsContexts` contains the matching context or `null` if it was not found or not set. You can use this field to implement an auto complete field for the existing topic name in the Save to Topic dialog box in your application.

TopicsContexts JSON

The `getTopicContexts` task returns `TopicContexts`, which contains a list of topic IDs and names.

```
{
    "topicsServiceContext": {
       TopicServiceContext object
    },
    “matchingTopicContext” : {
            "id": “number”,
            "name": "string"
     },
    "topicContexts": [
        {
            "id": “number”,
            "name": "string"
        }
    ]
}
```

Get a list of actions

Your application retrieves a list of actions for a topic available to the user by using the `getActionContexts` task that returns a list of action contexts with the action name and ID and the associated topic ID.

The result is sorted by name and includes the number of actions returned and the total number of values. You should add an option in your user interface to not select an action. The Oracle Empirica Signal application adds the **--none--** option to the top of the drop-down list. Optionally, you can set a start row and a chunk size.

ActionContexts JSON

The getActionContexts task returns a list of ActionContexts with action IDs and names and the associated topic ID.

```
{
    "topicsServiceContext": {
       TopicServiceContext object
    },
    "actionContexts": [
        {
            "id": “number”,
            "name": "string",
            "topicId": “number”
        }
    ]
}
}
```

Get a list of work teams

Your application can retrieve a list of work teams by using the `getWorkteamContexts` task.

This task returns a list of work team contexts with the work team name and ID. The result is sorted by name and includes the values returned and the total number of values.

When saving a topic to a topic workflow configuration that is set to **one and only one work team**, this task can be used to populate the list of work teams in the Save to Topic dialog box.

The `getWorkteamContexts` task returns `WorkteamContexts` with a list of work team names and IDs.

WorkteamContexts JSON

```
{
    "topicsServiceContext": {
       TopicServiceContext object
    },
    "workteamContexts": [
      {
        "id": “number”,
        "name": "string"
      }
    ]
}
```

Get a list of topic templates

Your application can retrieve a list of topic templates. The topic templates can be used to prepopulate new topic fields and actions.

To get a list of topic templates, call the `getTopicTemplateContexts` task, which returns a list of topic template contexts with the name and identifier of each topic template. This task can be used to populate a drop-down list or a selection list for the topic template in the Save to Topic dialog box. You should add an option in your user interface to not select a template. The Oracle Empirica Signal application adds the options **--none--** to the top of the drop-down list.

The result is alphabetically sorted by name and includes the values returned and the total number of values. Optionally, you can set a start row and a chunk size.

The `getTopicTemplateContexts` task returns `TopicTemplateContexts` with a list of topic template IDs and names.

TopicTemplateContexts JSON

```
{
    "topicsServiceContext": {
       TopicServiceContext object
    },
    "topicTemplateContexts": [
        {
            "id": “number”,
            "name": "string"
        }
    ]
}
```

Attach to an existing topic

Your application can create an attachment to an existing topic.

The topic argument in the `attachTopic` task contains the identifier of the topic. Set the **id** field of the `topicContext` argument to the ID of the topic to attach to.

AttachmentInput JSON

To create an attachment to an existing topic use the `AttachmentInput` value object.

```
{
    "serviceContext": {
        TopicServiceContext object
    },
    "topicContext":{
        “topicId” ; “number”
    },
    "attachment": {
    AttachmentInput object
    }
```

Attach to a new topic

Your application can create an attachment to a new topic.

The `TopicInput` value object has the required name field for the topic name and optional parameters. The `attachNewTopic` task creates an attachment to a new topic. For information about the `attachmentInput` argument, see Common value objects.

- **`projectName`**—Name to categorize the topic (required if `newProject` is true), with a maximum length of 255 characters.
- **`description`**—Optional description, with a maximum length of 2000 characters.
- **`workteams`**—A list of work team names.
- **`newProject`**—A boolean set to `True` to create new project.
- **`templateId`**—ID for the topic template.

The server creates the topic with a default initial state but no values are set for any topics fields. Topic names are not unique. If a topic is created with the same name as an existing topic, the user might see multiple topics with the same name in the Topic drop-down list. A user can display the Select Topic dialog box to distinguish between different topics with the same name by viewing other fields associated with the topic.

If specified, the **templateId** field should have the ID of the topic template from the user's selection and is available from the `TopicTemplateContext`.

If you chose not to support projects in your user interface or do not display the Project drop-down list, if the user creates a topic, you must set the **projectName** field to `Unassigned`.

TopicInput JSON

```
{
      "name" : "string",
      "description" : "string",
      "newProject" : "boolean",
      "projectName" : "string",
      "workteams" : ["string"],
   “templateId” : “number”
  }
```

To create a new topic without an attachment, do not supply the `"attachment"` attribute

attachNewTopic JSON

To create an attachment to a new topic:

```
{
    "serviceContext": {
        TopicServiceContext object
    },
    "topicInput": {
    TopicInput object
    },
```

```
    "attachment": {
    AttachmentInput object
    }
}
```

Attach to an existing action

Your application can create an attachment to an existing action by using the `attachAction` task.

The action argument contains the identifier of the action. Set the **id** field of the `actionContext` argument to the ID of the action to attach to and the **topicId** field to the topic ID for the action. For information on the `TopicServiceContext` and `attachmentInput` objects, see Common value objects.

attachAction JSON

```
{
    "serviceContext":{
        TopicServiceContext object
    },
    "actionContext" {
        "id": “number”,
        "topicId": “number”
    },
    "attachment": {
       AttachmentInput object
    }
```

Attach to a new action

Your application can create an attachment to an existing action by using the `attachAction` task.

The action input contains action information. Set the **topicid** field to the topic ID for the action. For information on the `attachmentInput` object, see Common value objects.

Your application can create an action without an attachment by not adding the `attachmentInput` attribute. The action will be created in the Initial state.

attachNewAction JSON

```
{
    "serviceContext":{
        TopicServiceContext object
    },
    "actionInput": {
        ActionInput object
    },
    "attachment": {
AttachmentInput object
```

```
    }
}
```

ActionInput value object

The `actionInput` value object specifies fields for the new action.

Fields for the new action:

- **Name** – Required. The name of the action.
- **actionTypeId** – Required. An action type ID.
- **description** – Description of the action.
- **topicId** – Required. The `topicId` on which the action will be created .

ActionInput JSON

```
{
        "name": "string",
        "actionTypeId": “number”,
        "description": "string",
        "topicId": “number”
}
```

Get field metadata

Your application can retrieve a list of all fields for topics, topic templates, work teams, or actions by using the `getFieldMetadata` task.

This task can be used to browse topics and choose a topic based on additional information, such as the assigned user, state, or any field, or to browse topic templates or work teams.

The `fieldContext` path values must be one of these: `topics`, `topicTemplates`, `workteams`, or `actions`.

This task returns the `FieldMetadata` value object with a list of field properties. The `TopicServiceContext` returns the total number of fields in the `totalRows` attribute.

FieldMetadata JSON

```
{
    "topicsServiceContext": {
    TopicServiceContext object
    },
    "fields": [
        {
            Field object
        }
     ]
}
```

Field value object

The `Field` value object describes a single field, which includes the field identifier, display label, type, whether the field is required, and whether the field can be presented to the user as a filter.

The field types are as follows.

`STRING, DATE, INTEGER, LONG, PROJECT, STATE, ACTION_TYPE`

Field JSON

```
{
    "id": "string",
    "label": "string",
    "type": "string",
    "filterable": “boolean”,
    "required": “boolean”
}
```

Get field values

Your application can retrieve a list of all field values for each topic, topic template, work team, or action by calling the `getFieldValues` task.

This task can be used to browse topics and choose a topic based on additional information, such as the assigned user, state, or to browse topic templates or work teams. Set the `fieldContext` argument to one of the enumeration values of `FieldContext`: TOPICS, TOPIC_TEMPLATES, WORKTEAMS, or ACTIONS. This task returns the `FieldValues` value object with a list of fields.

- For topics, the values reflect topics that are open and can be either viewed or attached to by the user.
- For work teams, the values reflect work teams associated with topics available to the user.
- For actions, the values reflect actions that can be viewed by the user.

Get filters

Your application can retrieve a list of fields that have been set up as filters and their values by using the `getFilters` task. The Filters object is returned from the `getFilters` task and is an ordered list of filter field names.

Your application might create a **Filter** drop-down list for each item in the `fieldNames` list. Set the label of the drop-down list to the string value of the `fieldNames` list. The Filters object also contains an array of filter values associated with each filter field name with the `label`, which is the text to display in the drop-down list, and the `value`, which is the field value associated with that label. The label can be the same as the value.

Filters JSON

```
{
    "topicsServiceContext": {
```

```
    TopicsServiceContext object
    },
    "fieldNames": [
        "string"
    ],
    "values": [
        {
            "values": [
                {
                    "label": "string",
                    "value": {
                        "fieldName": "string",
                        "stringValue": "string",
                        "type": "STRING"
                    }
                }
            ]
        }
    ]
}
```

Get a list of projects

Your application can retrieve a list of existing projects by using the `getProjects` task.

A project can be optionally specified when creating a topic. This task returns the `ProjectList` value object that always includes a value called **Unassigned** with ID **0**.

ProjectList JSON

```
{
    "topicsServiceContext": {
TopicServiceContent
    },
    "projects": [ “string”
    ]
}
```

Get a list of action types

Your application can retrieve a list of action types by using the `getActionTypes` task.

An action type is required when creating a new action. This task returns the `ActionTypeContexts` value object that always includes a value called **Generic** with ID **0**.

ActionTypeContexts JSON

```
{
    "actionTypeContexts": [
        {
            "id": “integer”,
            "name": "string"
```

```
        }
    ]
}
```

Get user info

Your application can view the topic permissions for a user by using the `getUserInfo` task.

Your application might hide or disable the **Save to Topic** link if the user cannot save to Oracle Empirica Topics.

TopicsUserInfo value object

The `TopicsUserInfo` task has the following fields:

- **canSavetoTopics**—True if a user can save to a topic. If true, your application can display the **Save to Topic** link.
- **canCreateTopic**—True if a user can create a topic. If true, your application can enable fields to create a topic in the Save to Topic dialog box.
- **canAccessMultipleWorkteams**—True if a user can select zero or more work teams for a new topic.
  If true, your application should not display the **Work Team** drop-down list in the Save to Topic dialog box.

  If false, then each new topic must be assigned to one work team. Your application must display a work team selector, such as a drop-down list, in the Save to Topic dialog box to assign the new topic to a work team.
- **canViewTopics**—Reserved for future use.
- **allowedAttachments**—A list of attachment types that a user can attach to a topic or action.

TopicsUserInfo JSON

```
{
    "username": "string",
    "canSaveToTopics": “boolean”,
    "canSaveToNewTopic": “boolean”,
    "canAccessMultipleWorkteams": “boolean”,
    "canViewTopics": “boolean”,
    "allowedAttachments": [
    “string”
    ]
}
```

Get topics service properties

Your application can retrieve the Oracle Empirica Topics server version and other properties by using the `getTopicsServiceProperties` task.

The `getTopicsServiceProperties` task returns the `TopicsServiceProperties` value object with the following fields:serverVersion—The version of the Oracle Empirica Topics server. If your API version is greater than the Oracle Empirica Topics server version, you might experience compatibility issues. attachMaxRows—Defines the maximum number of rows in a

TABLE attachment that your application can add. Your application is responsible for truncating the rows in a table to this value before sending it to the server.attachMaxMegabytes—Defines the maximum megabytes of any attachment. The topics service throws a TopicsServiceException if your application attempts to add an attachment greater than the attachMaxMegabytes value in megabytes.

TopicsServiceProperties JSON

```
{
    "serverVersion": “integer”,
    "attachMaxRows": “integer”,
    "attachMaxMegabytes": “integer”
}
```

Get topic workflow configurations

Your application can retrieve a list of topic workflow configurations by using the `getTopicWorkflowConfigurations` task.

You can view the topic workflow configurations for a user by using the `getTopicConfigs` task. Your application might hide or disable the **Save to Topic** link if the user cannot has no available workflow configuration.

TopicConfigContexts JSON

```
{
    "topicContexts": [
        {
            "id": “integer”,
            "name": "string"
        }
    ]
}
```

Stream file attachment

For an attachment of type FILE or IMAGE, clients must first stream files to the server and include a GUID in the `attachmentInput` parameter, and then call one of the attach tasks.

The attach tasks include `attachTopic`, `attachNewTopic`, `attachAction`, and `attachNewAction`.

The task uses a query parameter to specify the resulting `guid`. The `guid` parameter is required. The body of the message must be a `BINARY_INPUT_STREAM` containing the file contents.

Exceptions

Exceptions are thrown when the Topics service detects a user error or an internal error.

Topics service tasks may throw the following exceptions:

- **InvalidArgumentException**—Thrown for internal errors if you define invalid information in a field or argument. One example is passing a `TopicsServiceContext` argument with a value of null. Another example is requesting actions for a topic ID of **-5**. Topic IDs must be a number greater than or equal to 0.
- **TopicsServiceException**—Thrown for errors other than `InvalidArgumentException`. This includes user input errors. Another example is a semantic error such as getting actions contexts for a topic with an ID that does not exist, or for a topic that was deleted or closed.

These exceptions contain a **message** field.

The code field contains the HTML code for the error and the status contains the html status:

```
400 BAD_REQUEST
```

Other HTML statuses, such as SUCCESS, or AUTHORIZATION FAILURE, may be obtained from the response.

TopicsServiceException value object

```
{
    "errors": [
        {
            "message": string",
            "status": "string",
            "code": “integer”
        }
    ]
}
```

Table 3–3 TopicsServiceException error codes

Error Code	Description
TOPIC_ERROR_INTERNAL	Internal error.
TOPIC_ERROR_NO_SERVICE	Communication error with the topic server.
TOPIC_ERROR_TOPIC_NAME_REQUIRED	*attachNewTopic* task called with the *topicInput* parameter name field not set.
TOPIC_ERROR_ATTACHMENT_NAME_ REQUIRED	*attachTopic*, *attachNewTopic*, or *attachAction* task called with the *attachmentInput* parameter name field not set.
TOPIC_ERROR_PROJECT_NAME_REQUIRED	*attachNewTopic* task called with *topicInput parameternewProject* field set but *projectName* field not set.
TOPIC_ERROR_TOPIC_CLOSED	*attachTopic* task called with the *topicContext* parameter field id set to a closed topic.
TOPIC_ERROR_ACTION_CLOSED	*attachAction* task called with the *actionContext* parameter field id set to a closed action.
TOPIC_ERROR_TOPIC_DELETED	*attachTopic task called with the topicContext parameter field id set to a deleted topic.*
TOPIC_ERROR_ACTION_DELETED	*attachAction task called with the actionContext parameter field id set to a deleted action.*

Error Code	Description
TOPIC_ERROR_UNAVAILABLE_TOPIC	*attachTopic task called with the topicContext parameter field id set to a topic that is not available.*
TOPIC_ERROR_UNKNOWN_ACTION	attachAction task called with the actionContext parameter field id set to an action that is not available.
TOPIC_ERROR_UNKNOWN_TOPIC	Reserved for future use.
TOPIC_ERROR_ATTACHMENT_NAME_LENGTH	*attachTopic*, *attachNewTopic*, or *attachAction* task called with the attachment name length greater than the maximum length of 255 characters.
TOPIC_ERROR_TOPIC_DESCRIPTION_LENGTH	*attachNewTopic* task called with the topic description length greater than the maximum length of 2000 characters.
TOPIC_ERROR_ATTACHMENT_DESCRIPTION_LENGTH	*attachTopic*, *attachNewTopic*, or *attachAction* task called with the attachment description length greater than the maximum length of 2000 characters.
TOPIC_ERROR_TOPIC_NAME_LENGTH	*attachNewTopic* task called with the topic name length greater than the maximum length of 255 characters.
TOPIC_ERROR_PROJECT_LENGTH	*attachNewTopic* task called with a new project name length greater than the maximum length of 255 characters.
TOPIC_ERROR_URL_REQUIRED	*attachTopic*, *attachNewTopic*, or *attachAction* method called with the attachment type of URL and with the *AttachmentInput urlAddress* empty.
TOPIC_ERROR_URL_LENGTH	*attachTopic*, *attachNewTopic*, or *attachAction* task called with the attachment type of URL and the *AttachmentInput urlAddress* greater than the maximum length of 2000 characters.
TOPIC_ERROR_URL_INVALID	*attachTopic*, *attachNewTopic*, or *attachAction* task called with the attachment type of URL and an invalid format for the *AttachmentInput urlAddress*.
TOPIC_ERROR_NOTE_REQUIRED	*attachTopic*, *attachNewTopic*, or *attachAction* task called with the attachment type of Note with the *AttachmentInput* description empty.
TOPIC_ERROR_NOTE_LENGTH	*attachTopic*, *attachNewTopic*, or *attachAction* task called with the attachment type of Note and the *AttachmentInput* description greater than the maximum of 2000 characters.
TOPIC_ERROR_ATTACHMENT_IMAGE_TABLE_NOT_ALLOWED	*attachTopic*, *attachNewTopic*, or *attachAction* task called when *TopicsUserInfo allowedAttachments* does not contain IMAGE or TABLE.
TOPIC_ERROR_ATTACHMENT_URL_NOT_ALLOWED	*attachTopic*, *attachNewTopic*, or *attachAction* task called when *TopicsUserInfo allowedAttachments* does not contain URL.
TOPIC_ERROR_ATTACHMENT_FILE_NOT_ALLOWED	*attachTopic*, *attachNewTopic*, or *attachAction* task called when *TopicsUserInfo allowedAttachments* does not contain FILE.
TOPIC_ERROR_ATTACHMENT_NOTE_NOT_ALLOWED	*attachTopic*, *attachNewTopic*, or *attachAction* task called when *TopicsUserInfo allowedAttachments* does not contain NOTE.

4

Secure development guidelines

Security guidelines

Follow these guidelines for secure development.

Transport-level security

The Oracle Empirica Topics web service client must use HTTPS connection to secure all data communication with the Oracle Empirica Topics web service.

Message-level security

The Oracle Empirica Topics web service client must use WS SECURITY and either a user name token policy or a bearer token.

- The user name token policy must have an Oracle Empirica Topics web service user name and password.
- The bearer token must contain the oAuth token from your oAuth provider.

These policies are used to connect securely to the Oracle Empirica Topics web service and to authenticate the client for each API call.

Access control security

The Oracle Empirica Topics web service client must include a user name in the input field **TopicsServiceContext.username** for all API calls except for `getTopicsServiceProperties`. The `getTopicsServiceProperties` API call is available to any user.

The `getTopicsServiceProperties` API call is available to any user.

For any calls specific to a topic workflow configuration, the web service client must also include a *twcId* in the input field.

> **Note**
>
> Each API call is protected by the same work team membership, roles, permissions, and topic configuration that are defined via Oracle Empirica Signal administration for that user name, if any.

Request parameter validation

The Oracle Empirica Topics web service validates API input fields for content and size.

- The Oracle Empirica Topics web service validates attachment size based on the size requirements for attachments as defined in the Oracle Empirica Signal Site Properties.
- If validation fails, the Oracle Empirica Topics web service call returns an exception and does not perform the API call.

5

XML attachment tables

About XML attachment tables

The XML for the attachment type TABLE contains metadata for each of the columns, including the type, notes, and the rows of data.

Each column has the format specified in the <DBType> element. The integer value for each data type is determined by the integer values in Java and their respective SQL data types:

Table A–1 Data types

Value	Data Type	Description
2	NUMERIC	Specifies one of the following: • An integer, such as 25. • A double formatted value, such as 64-bit IEEE 754. • A floating point number, with up to 6 digits after the decimal point, such as 25.123456 or 0.123.
12	VARCHAR	Specifies a string.
91	DATE	Specifies a date without time in a string using the ISO8601 format, such as 1997-07-16 in the format YYYY-MM-DD.
93	TIMESTAMP	Specifies a timestamp with the date and time in a string using the ISO8601 format, such as 1997-07-16T19:20:30Z in the format YYYY-MM-DDThh:mm:ssZ where: • The letter T separates the date and time, including hours, minutes, and seconds. • The letter Z follows the time and represents the UTC time zone. Dates are formatted in the UTC time zone (Coordinated Universal Time).

XML—attachment type: TABLE

The XML code illustrates the format for the XML that is streamed for TABLE attachments.

```
<TableData>
 <MetaData>
  <TableTitle>[String Value]</TableTitle>
  <MultiRowHeading>Y|N</MultiRowHeading>
  <Column name="[String Value]">
   <Label>[String]</Label>
   <DBType>2|12|91|93</DBType>
   <ToolTip>[String Value]</ToolTip>
   <Visible>Y|N</Visible>
  </Column>
  <Column name="[String Value]">
   <Label>[String]</Label>
   <DBType>2|12|91|93</DBType>
   <ToolTip>[String Value]</ToolTip>
   <Visible>Y|N</Visible>
  </Column>
  <Column name="[String Value]">
   <Label>[String]</Label>
   <DBType>2|12|91|93</DBType>
   <ToolTip>[String Value]</ToolTip>
   <Visible>Y|N</Visible>
  </Column>
  <Notes>
   <NotesDetail title="[String Value]">[HTML Formatted String Value]</
NotesDetail>
   <NotesDetail title="[String Value]">[HTML Formatted String Value]</
NotesDetail>
   <NotesDetail title="[String Value]">[HTML Formatted String Value]</
NotesDetail>
  </Notes>
 </MetaData>
 <Row>
   <Column name="[String Value]">[String Value]</Column>
   <Column name="[String Value]">[String Value]</Column>
   <Column name="[String Value]">[String Value]</Column>
 </Row>
 <Row id="[String Value of Row Number]">
   <Column name="[String]">[String Value]</Column>
   <Column name="[String]">[String Value]</Column>
   <Column name="[String]">[String Value]</Column>
 </Row>
</TableData>
```

The <Metadata> contains the column definitions, an optional title or header above the table and notes. The contents of notes for tables are determined by <Notes> and <NotesDetail> XML elements. The XML must contain all these elements except for <NotesDetail>.

The <Row> elements contain the data for the table, and optional notes for the table. See Figure 2-11, Attachment—View, and Figure 2-12, Attachment—View and Show Notes. The rows of

the table are determined by XML children of the <TableData> element that is streamed with the attachment.

1. <MetaData>—Define <MetaData> as a child of <TableData> and define the following child elements of <MetaData>.

   a. <TableTitle>—Define with optional text to be displayed above the table for additional information about the data, such as a filter. You must set the content of this element to empty <TableTitle/> if there is no text to display.

   b. <MultiRowHeading>—Define with content of **Y** or **N**. Set the value to **Y** to display two rows of column headers when the attachment is displayed. The first row displays the first word of each column label and the second row displays the rest of each label. For example, use this if you have two rows of columns in your table where the first row has the same word across more than one column and the second row has a distinguishing label.

   The following column headers appear if MultiRowHeader is set to **Y**.

All	All	Serious	Serious
N Since 2010	EBGM 2012	N Since 2010	EBGM 2012

If the value to set to **N**, then only one row of column headers is displayed.

All	All	Serious	Serious
N Since 2010	EBGM 2012	N Since 2010	EBGM 2012

   a. <Column>—Specify a list of <Column> header elements where there is one column element for each column header in your attachment. Set the name attribute to an arbitrary string that is used to match the same column name attribute in the <Row> elements that contain the row data.

   - <Label>—Set the <Label> content to the text to display in the column header.
   - <DBType>—Set the <DBType> content to the column type integer value as described in Table A–1. This value is used for formatting values in the table such as left or right justification of the text. Numbers are right aligned. All other types are left aligned.
   - <ToolTip>—Set the <ToolTip> content to the text to display for the tooltip on the column name or <ToolTip/> if there is no tooltip.
   - <Visible>—Set the <Visible> content to the default visibility of the column. Set to **Y** to show this column by default or **N** to hide this column. Users can select the **Columns** link to show or hide columns when viewing the attachment.

   a. <Notes>—Define children for the <Notes> elements to display the **Show Notes** link above the table when the attachment is displayed. If you do not have notes, then you must specify an empty notes element <Notes/> and the **Show Notes** link will not be displayed in the attachment.
   Notes are displayed as a table with two columns. The <Notes> tag has one or more optional <NotesDetail> child elements. Define the title attribute to the text for the first column of the note table such as a note row label. Define the content of the <NotesDetail> for text of the second column, such as a note row value or <NotesDetail/> if there is no text for this row. The <NotesDetail> content can optionally contain HTML tags for formatting and can contain HTML <table> related tags to embed a table in the second column of the notes.

2. <Row>—Define a list of <Row> elements with one <Row> for each row in your table. For each <Row> , define the following child elements.

   a. <Column>—You must define one <Column> child element for each <Column> element that is a child of the <Metadata> element, and define the same name attribute. This name can be any unique name for the column. Set the content of this element to the text to be displayed in the content of this column's row in the table.

   If the column name is set to **reviewed** or **excluded**, which are used by Empirica applications, then any data in those columns are left aligned regardless of the DBType attribute. The values of the rows for these column names are displayed as **YES** if set to 1 or **NO** if set to **0**.

6 Glossary

action

An activity that contributes to a topic, such as performing research, completing project management milestones, or collecting supporting documentation. An action includes a workflow and state mechanism with user assignments.

For example, in the Oracle Empirica Signal application, default fields for an action include name, description, state, assigned to user, planned completion date, and action completion date.

An action name does not need to be unique but has a unique ID.

attachment

Text, XML, or binary data associated with an action or a topic. An attachment may include comments and custom fields.

An attachment name is not required to be unique.

attachment type

Attachment types include:

- **File**
- **Image**—A ZIP file containing one or more files with the JPG, JPEG, PNG, or GIF extension, optionally one or more `notes.txt` files that contain associated notes encapsulated in an HTML <table> tag and optionally one or more `*.txt` files with HTML formatted strings. TXT files can be used for titles or annotations in the attachment. When rendered in a topic, an image attachment is converted to a PDF comprised of the files in the same order as stored in the ZIP file. The PDF is rendered when the user views the attachment.
- **Note**—A text note, with a maximum length of 2000 characters.
- **Table**—An XML representation of a table. See About XML attachment tables for more information.
- **URL**—A URL address in the form `http[s]://xxx[:port][/yyy.zzz]`, with a maximum length of 2000 characters.

field metadata

Information about a field name, type, visibility, necessity, and whether the field can be used as a filter. Topics, topic templates, actions, and work teams have field metadata defined in the workflow configuration.

field value

An instance of the field metadata for a topic, topic template, action, or work team (for example, a list of the field values for a topic).

project

A named category. A topic can optionally be grouped into at most one project.

state

A named part of the topic workflow. A user can edit a topic or action and transition it between states in the workflow.

topic

A means of organizing and tracking issues identified from various applications. Reference materials, including result tables, graphs, reports, and external documents related to the subject, can be stored with the topic.

A topic name is not required to be unique but has a unique ID.

topic template

A topic template has prepopulated values that are applied to a new topic. Optionally, a topic template includes topic field values and action field values and can be used when saving to a new topic.

topics service

The Oracle Empirica Topics service that handles the Oracle Empirica Topics API methods described in this document. This is a web service configured as part of the Oracle Empirica Signal application.

topics service context

A parameter to every Oracle Empirica Topics task that includes context information such as the username and information for sorting, filtering, and paging.

user name

A username is passed from the client to the server to validate that the user can access or save to particular topics or actions that are associated with work teams.

work team

A work team is a subset of users from an Oracle Empirica Signal login group, which is a grouping of users that collaborate on a topic. There can be multiple work teams within the same login group, and the same user can be in one or multiple work teams.

Topics that are made visible to a work team can be viewed by, or assigned to, any of its members depending on the user's work team permissions.

workflow configuration

A workflow configuration contains the topic and action configuration and includes topic and action states, topic and action field metadata, and configuration settings.

Change log

Date	Part number	Description
November 2025	G39969-01	Guide describing the addition of REST APIs in the 2025.4.01 release for custom programming.