Oracle® Health Sciences Empirica Topics

API Guide

Release 8.0

E55424-01

September 2014

Oracle Health Sciences Empirica Topics API Guide, Release 8.0

E55424-01

Contents

3 API

A XML attachment tables

Glossary

List of Figures

List of Tables

Preface

The preface includes the following sections:

- Audience
- Documentation accessibility
- Related documents
- Conventions

Audience

This document is intended for programmers who want to integrate their proprietary application with Empirica Topics. Programmers can use the Empirica Topics Services API to attach artifacts from their proprietary application to topics.

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`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related documents

For more information, see the following documents in the Oracle Health Sciences Empirica Signal Release 8.0 documentation set:

- *Oracle Health Sciences Empirica Signal and Topics User Guide*
- *Oracle Health Sciences Empirica Signal Release Notes*
- *Oracle Health Sciences Empirica Signal Known Issues*
- *Oracle Health Sciences Empirica Topics Reporting and Oracle Business Intelligence Configuration Guide*
- *Oracle Health Sciences Empirica Signal Installation Guide*
- *Oracle Health Sciences Empirica Signal Secure Configuration Guide*

- *Oracle Health Sciences Empirica Signal Third Party Licenses and Notices*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Web service configuration

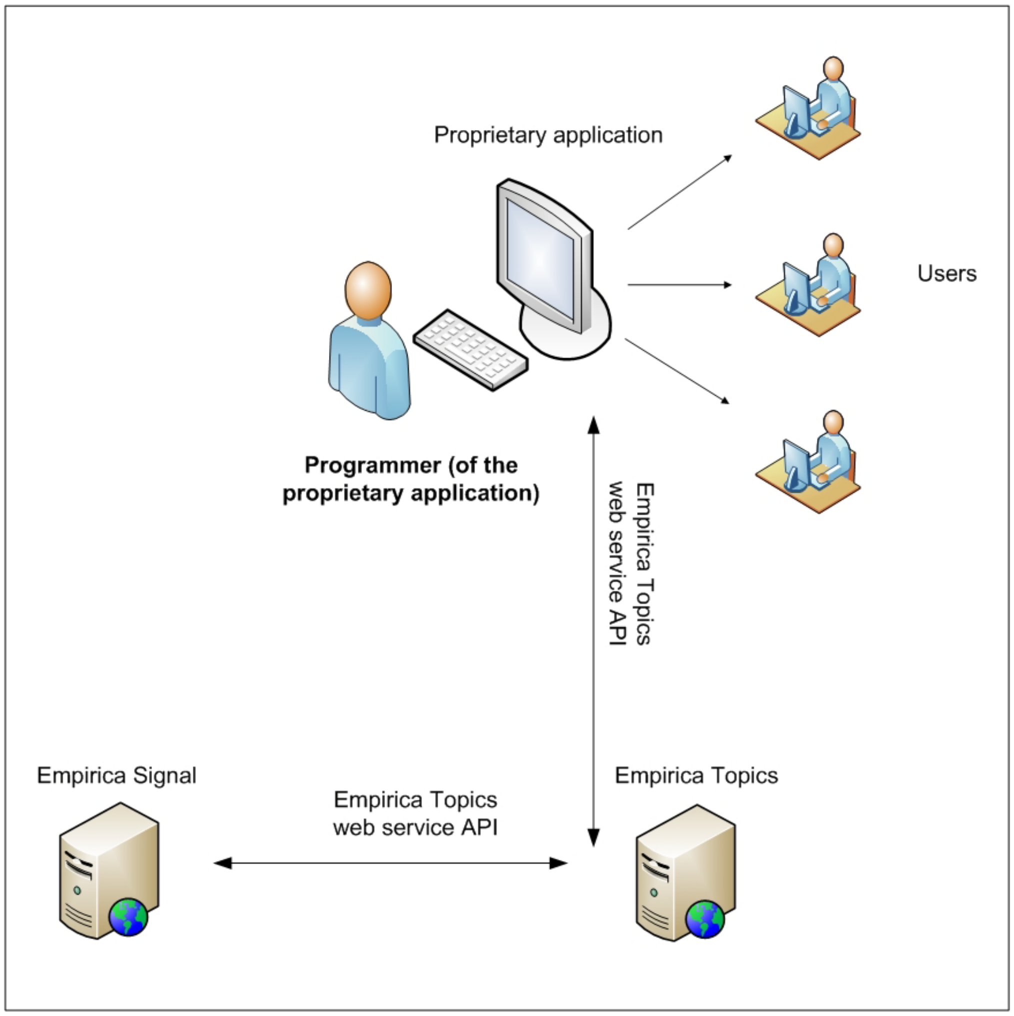
This chapter includes the following sections:

- Overview
- Configuring users
- Security
- WSDL
- Connecting to the Topics Web Service

Overview

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

Figure 1–1 API overview


Proprietary application
Users
Programmer (of the proprietary application)
Empirica Topics web service API
Empirica Signal
Empirica Topics web service API
Empirica Topics

In your application, you must add integration points that implement the Empirica Topics web service API, which is available using the Topics WSDL.

In this document, code snippets are displayed using Java. For a .NET application, C# techniques are comparable.

Note: Empirica Topics is installed as part of the Empirica Signal application and includes:

- The Empirica Topics web service that implements the Empirica Topics API for Save to Topic in your proprietary application.
- The Topics tab in Empirica Signal for end users to view and edit their topics.
- Topics administration in Empirica Signal for administrators to manage topic workflow configurations, work teams, user accounts and their permissions.

Configuring users

The 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 Topics permissions and work teams on the Empirica Signal server. The Topics web service validates these 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 more information, see Chapter 3, 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 WebLogic administrator must create a user for exclusive use by your application and the Topics web service. The Topics web service username and password credentials are provisioned on the 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 Topics server validates access to the Topics web service using these credentials. The Topics server can return an exception if the web service credentials are not correct or the Username Token Policy has not been set. The Topics web service will then not be available to your application.

WSDL

You import the Empirica Topics WSDL to generate Java or .NET proxy classes in order to access the Topics Service API. For more information on the WSDL, see the Empirica Signal distribution file, `Signal_Install.tar.gz` and the following files:

- `Topics_Service/EmpiricaTopicsService.wsdl`
- `Topics_Service/EmpiricaTopicsService_schema1.xsd`

Alternatively, your application may import from an active Empirica Topics server such as `http://`*<hostname>*`:`*<port>*`/Signal/ws/topicsService?wsdl` to generate a stub for the Topics web service.

Connecting to the Topics Web Service

To connect your application to the 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 Topics web service such as

  `https://`*<hostname>*`:`*<port number>*`/Signal/ws/topicsService`

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

- Http chunk size when sending large attachments (for Java only)

  This property specifies the maximum chunking size that the web service uses when sending attachments.

- Username

  This is the Topics service username.

- Password

  This is the Topics service password.

You also need to enable the *MTOMFeature* in your proprietary application when calling Topic service APIs that send attachments.

Java integration

For Java applications, set the *MTOMFeature* for your web service. Also set up WS SECURITY and the username token policy by defining the *SecurityPoliciesFeature* for your web service.

The following example demonstrates how to connect to the Topics web service for each call:

```
// Instantiate the Topics web service proxy
EmpiricaTopicsService topicsService = new EmpiricaTopicsService(
  new URL("https://<hostname>:<port number>/Signal/ws/topicsService?wsdl"),
  new QName("http://oracle.hsgbu.topics.com", "EmpiricaTopicsService"));
BindingProvider bindingProvider = (BindingProvider)topicsService;
// Set the endpoint address for the Topics web service
bindingProvider.getRequestContext().put(
  BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
  "https://<hostname>:<port number>/Signal/ws/topicsService");
// Set the username token policy
SecurityPoliciesFeature securityFeatures = new SecurityPoliciesFeature(
  new String[]{"oracle/wss_username_token_over_ssl_client_policy"});
// Include MTOM if this API call sends attachments
topicsService.getEmpiricaTopicsPort(new MTOMFeature(), securityFeatures);
// Set the topics username and password
bindingProvider.getRequestContext().put(
  BindingProvider.USERNAME_PROPERTY, <topics service username>);
bindingProvider.getRequestContext().put(
  BindingProvider.PASSWORD_PROPERTY, <topics service password>);
// Set a maximum chunk size when calling an API that streams attachments
bindingProvider.getRequestContext().put(
  "com.sun.xml.ws.transport.http.client.streaming.chunk.size", 8192);
```

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

.NET integration

If your application uses .NET, you specify similar properties to the Java integration to connect to the Topics web service, but for .NET, the calls to connect are different. For example, your application will pass the Topics service URL as a parameter to connect to the Signal web service.

You must use standard .NET practices to integrate with WS Security and the Username Token Policy.

2 Use cases

This chapter includes the following sections:

Implementing the Save to Topic features

Using the Topics tab in Empirica Signal, an end user can create, display, edit, and move topics through a workflow. Topics can have attachments such as files, graphs, or tables. Topics can also contain one or more actions which 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 Empirica Signal administrator defines topic workflow configurations and configures work teams and their permissions.

Your proprietary application can use the 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/or actions.

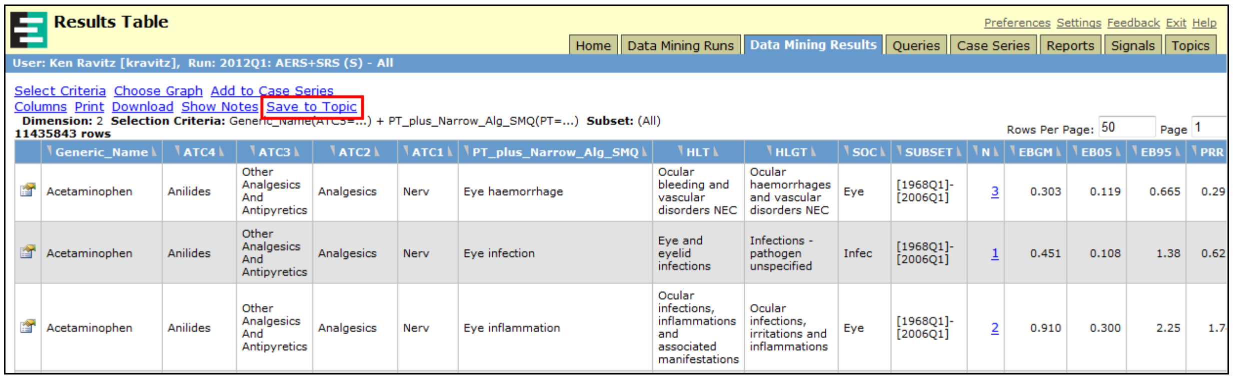
The use cases in this chapter describe a sample implementation by Empirica Signal of the Save to Topic features and associated web services calls. You can integrate with Save to Topics in your user interface in a way that is appropriate for your proprietary application.

Save to Topic link

Your application can incorporate a **Save to Topic** link into its pages, similar to how 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 Topics web service. The Topics web service initializes the values and saves an attachment containing the application's data to the specified topic or action.

The following example (Figure 2–1) shows integration points of the Empirica Signal application with Empirica Topics. You can add the **Save to Topic** link to any page in your application with graphs or tables, or with pages that can be rendered as a file. For example, you can render the contents of a page as a PDF file, and save the PDF file to a topic.

Figure 2–1 Save to Topic link in the Empirica Signal Data Mining Results Table



User: Ken Ravitz [kravitz], Run: 2012Q1: AERS+SRS (S) - All

Select Criteria Choose Graph Add to Case Series
Columns Print Download Show Notes Save to Topic

Dimension: 2 **Selection Criteria:** Generic_Name(ATC3=...) + PT_plus_Narrow_Alg_SMQ(PT=...) **Subset:** (All)
11435843 rows

Rows Per Page: 50 Page 1

	Generic_Name	ATC4	ATC3	ATC2	ATC1	PT_plus_Narrow_Alg_SMQ	HLT	HLGT	SOC	SUBSET	N	EBGM	EB05	EB95	PRR
	Acetaminophen	Anilides	Other Analgesics And Antipyretics	Analgesics	Nerv	Eye haemorrhage	Ocular bleeding and vascular disorders NEC	Ocular haemorrhages and vascular disorders NEC	Eye	[1968Q1]-[2006Q1]	3	0.303	0.119	0.665	0.29
	Acetaminophen	Anilides	Other Analgesics And Antipyretics	Analgesics	Nerv	Eye infection	Eye and eyelid infections	Infections - pathogen unspecified	Infec	[1968Q1]-[2006Q1]	1	0.451	0.108	1.38	0.62
	Acetaminophen	Anilides	Other Analgesics And Antipyretics	Analgesics	Nerv	Eye inflammation	Ocular infections, inflammations and associated manifestations	Ocular infections, irritations and inflammations	Eye	[1968Q1]-[2006Q1]	2	0.910	0.300	2.25	1.7

Topics methods

To implement the **Save to Topic** link, your application invokes the *getUserInfo* method. 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.getUserInfo().isCanSavetoTopic()
```

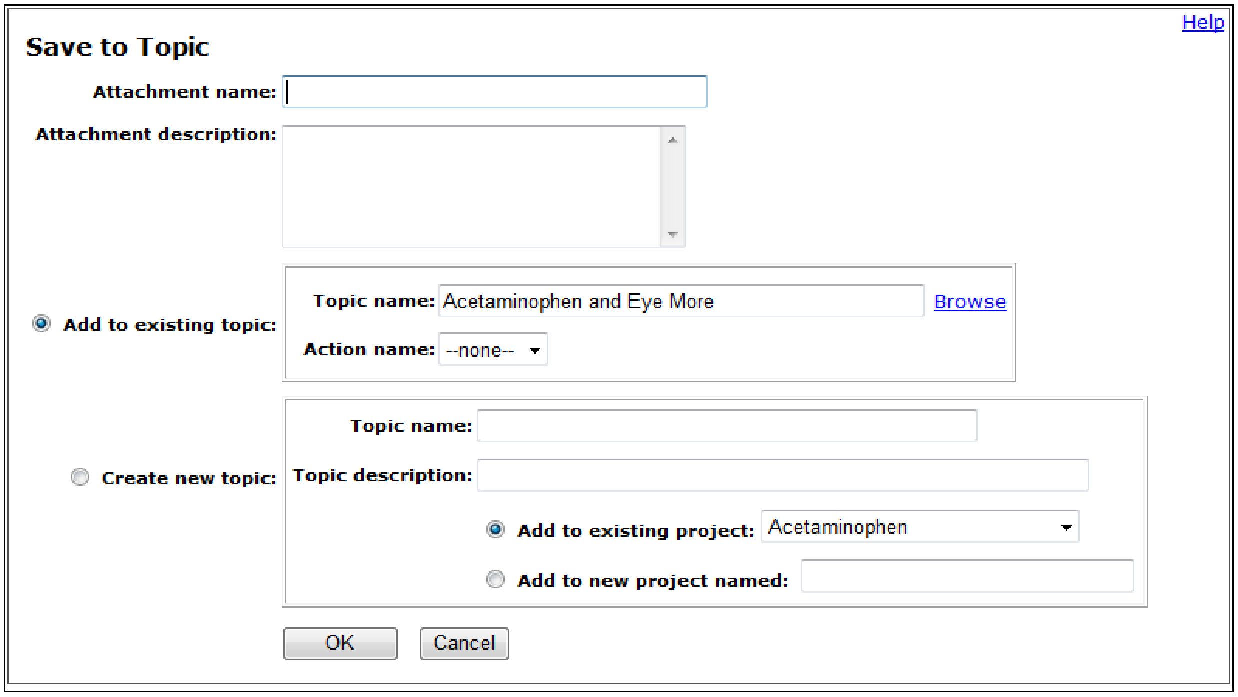
The user must have the same username in your application and in Empirica Topics. *TopicsUserInfo.getUserInfo()* gives a TopicsServiceException if the username from your application is not found.

Save to Topic dialog box

When the user clicks **Save to Topic**, the Save to Topic dialog box appears. In the Save to Topic dialog box, 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 2–2) and can vary based on your application's design.

Figure 2–2 Save to Topic dialog box in Empirica Signal


Save to Topic

Help

Attachment name:

Attachment description:

[✓ Add to existing topic:] Topic name: Acetaminophen and Eye More Browse
Action name: --none--

Create new topic: Topic name:
Topic description:
[✓ Add to existing project:] Acetaminophen
Add to new project named:

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.

Topics methods

Your application invokes the following Topics methods to render the Save to Topic dialog box:

- *getUserInfo*—Gets the user privileges. The method returns the TopicsUserInfo value object which contains the field *CanSaveToNewTopic*. If this field is false, you can disable the section for creating topics.
- *getTopicContexts*—Gets all topics to which the user can save an attachment and populates the existing topics drop-down list.

  Note: Empirica 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.

Note: Empirica 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 method is used when creating a topic.

Note: Empirica 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 Empirica applications populate the project drop-down list, they combine the project names returned from the 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.

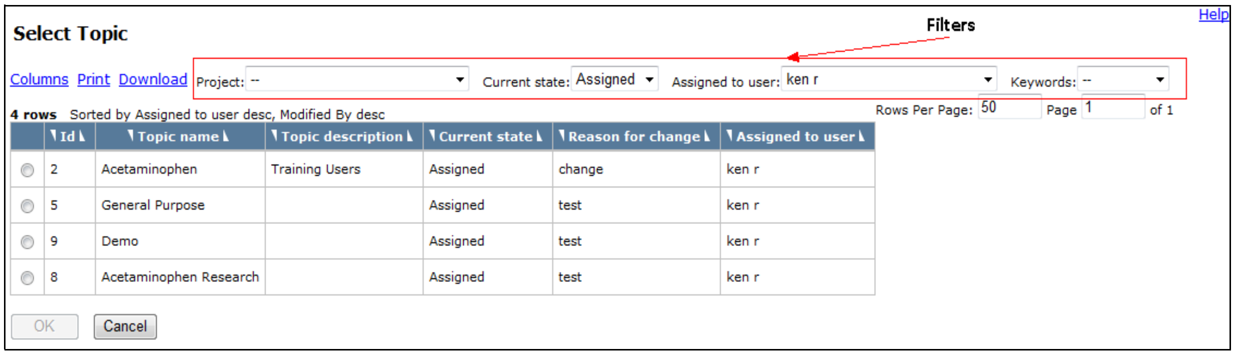
Select Topic dialog box

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

Your application should create a Select Topic dialog similar to the example (Figure 2–3).

Figure 2–3 Select Topic dialog box in Empirica Signal



Project: --   Current state: Assigned   Assigned to user: ken r   Keywords: --

4 rows   Sorted by Assigned to user desc, Modified By desc   Rows Per Page: 50   Page 1 of 1

	Id	Topic name	Topic description	Current state	Reason for change	Assigned to user
○	2	Acetaminophen	Training Users	Assigned	change	ken r
○	5	General Purpose		Assigned	test	ken r
○	9	Demo		Assigned	test	ken r
○	8	Acetaminophen Research		Assigned	test	ken r

Topics methods

Your application invokes the following Topics methods 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.

Save to topic with topic templates

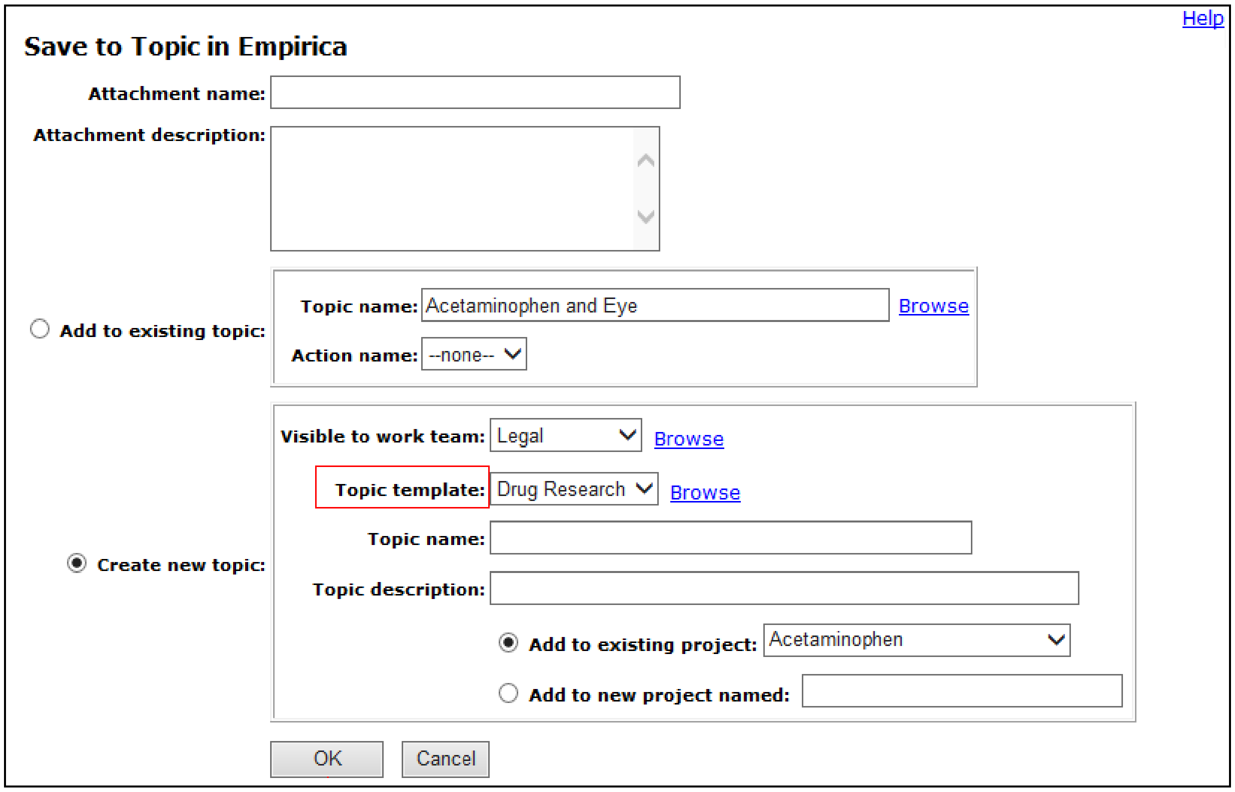
If topic templates have been created by an Empirica Topics administrator, and they are visible to a work team to which the user belongs, then the **Topic template** drop-down list should appear in the Save to Topic dialog box.

If the user selects a topic template from the **Topic template** drop-down list when creating a new topic, the new topic is populated with the following from the template:

- Topic field values for the initial topic state
- Actions and their field values for the initial action state

If the user does not select a template from the **Topic template** drop-down list when creating a new topic, none of the fields of the new topic will have pre-defined values and no actions will be created automatically for the topic.

Figure 2–4 Save to Topic dialog box in Empirica Signal when topic templates are available to the user



Attachment name:

Attachment description:

( ) Add to existing topic:
Topic name: Acetaminophen and Eye
Action name: --none--

(●) Create new topic:
Visible to work team: Legal
Topic template: Drug Research
Topic name:
Topic description:
(●) Add to existing project: Acetaminophen
( ) Add to new project named:

Topics methods

Your application invokes the following Topics method to render the **Topic template** drop-down list:

- *getTopicTemplateContexts*—Gets the topic templates visible to at least one work team to which the user belongs.

Save to topic and work teams

An Empirica Topics administrator can configure a topic workflow configuration such that the topics are visible to one of the following:

- One and only one work team

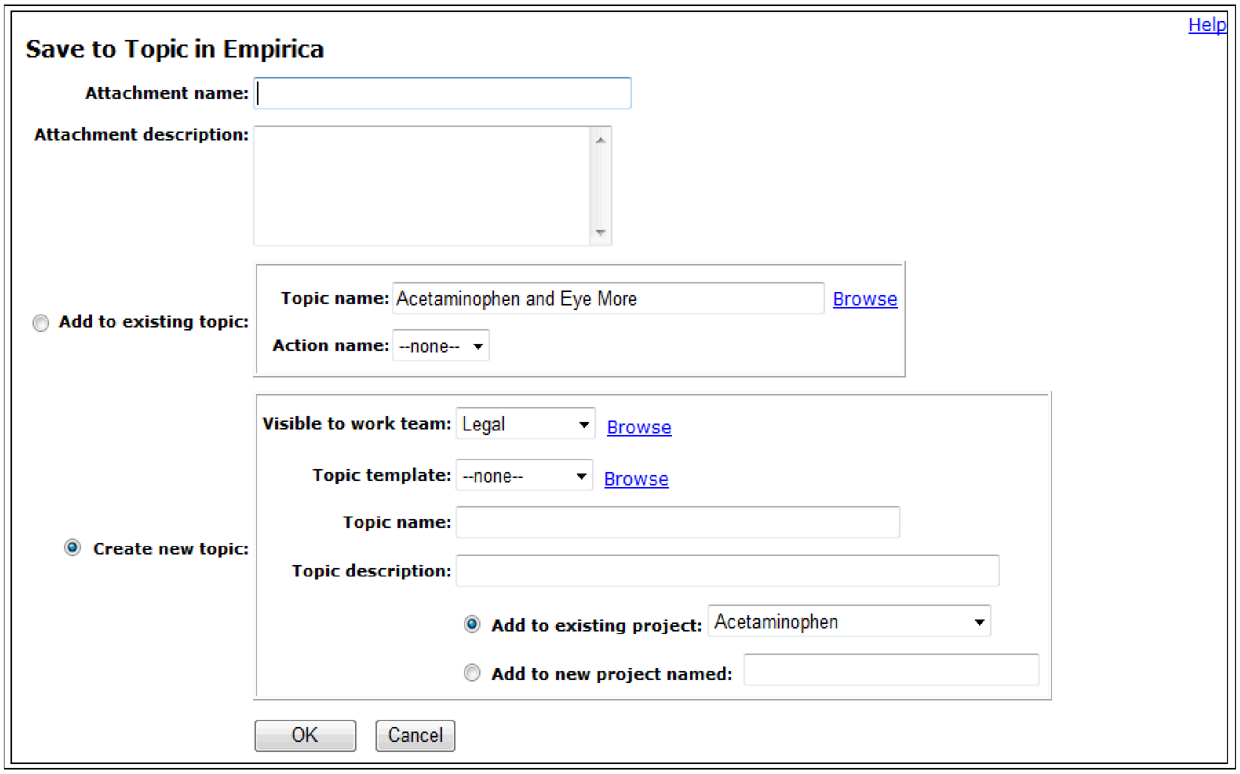
  or

- Zero, one, or more 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 Empirica Topics application and publish it by assigning it to one or more work teams.

Figure 2–5 Save to Topic dialog box in Empirica Signal configured for one and only one work team



Save to Topic in Empirica

Help

Attachment name:

Attachment description:

( ) Add to existing topic:
Topic name: Acetaminophen and Eye More Browse
Action name: --none--

(●) Create new topic:
Visible to work team: Legal Browse
Topic template: --none-- Browse
Topic name:
Topic description:
(●) Add to existing project: Acetaminophen
( ) Add to new project named:

Topics methods

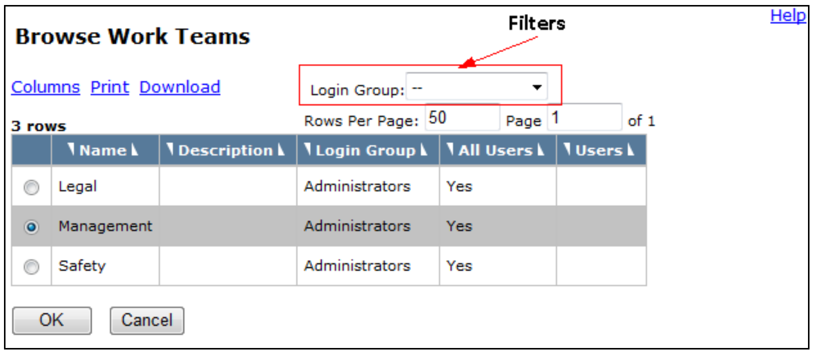
Your application invokes the following Topics method to render the **Visible to work team** drop-down list:

- *getWorkteamContexts*—Gets the work teams based on the user's work team permissions.

Browse work teams

In the Save to Topic dialog box, users can browse work teams. The user clicks **Browse** to navigate to the Browse Work Teams dialog box (Figure 2–6) and can select a work team. When the user selects a work team and clicks **OK**, the work team appears in the **Visible to work team** drop-down list.

Figure 2–6 Browse Work Teams dialog box in Empirica Signal



Browse Work Teams

Login Group: --

3 rows

Rows Per Page: 50 Page 1 of 1

	Name	Description	Login Group	All Users	Users
○	Legal		Administrators	Yes	
●	Management		Administrators	Yes	
○	Safety		Administrators	Yes	

Topics methods

Your application invokes the following Topics method 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 method to FieldContext.WORKTEAMS.

Saving attachments

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

Topics methods

Your application invokes the following Topics methods to save the attachment when the user clicks **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.

Streaming attachments

If your application uses Java, use the JAX-WS infrastructure and DataHandler to stream the data to the topics server.

If your application uses .NET, first call the Topics service API *streamFileAttachment()*, and then call one of the three Topics web service calls to attach the file to a topic or an action.

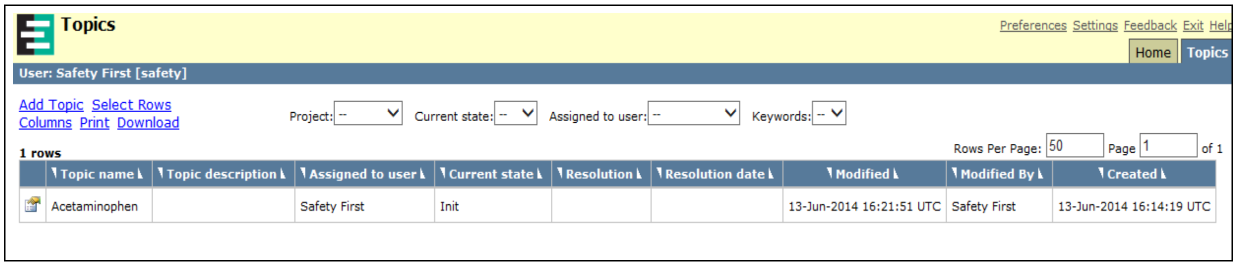
Viewing topics

The Topics page shows a table of all the topics that are accessible by the user. For example, the user can edit the first topic **Acetaminophen** using the action menu in the first column of the table (Figure 2–7).

In Empirica Topics, a user can:

- View or edit topics and actions.
- Add attachments such as files and URLs.
- View attachments added using the **Save to Topic** interface.
- View the source information associated with an attachment.
- Change the topic state to move the topic through the workflow.

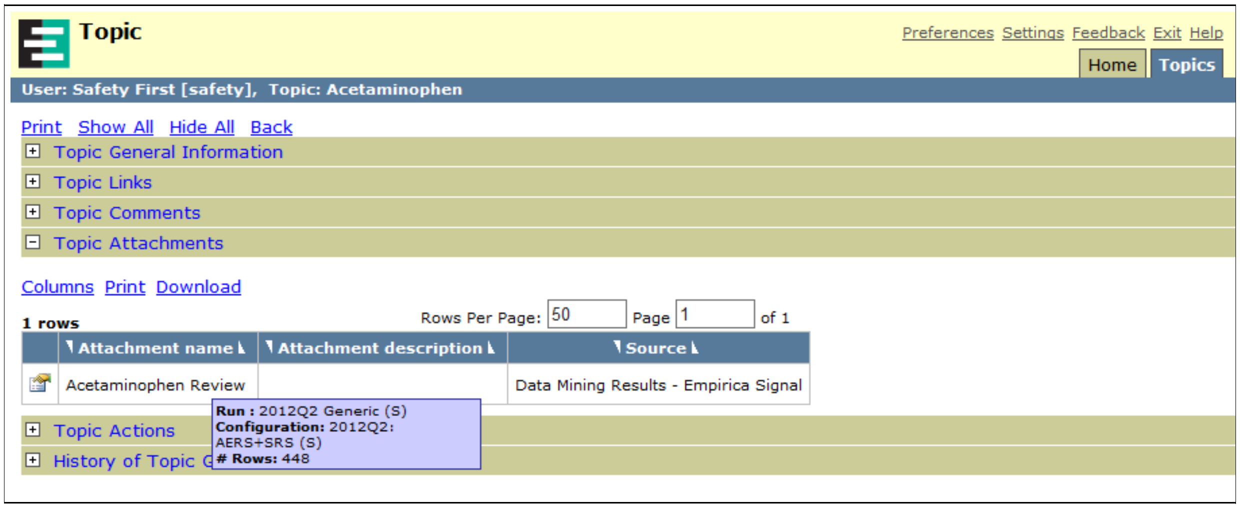
Figure 2–7 Topics tab in Empirica Signal



	Topic name	Topic description	Assigned to user	Current state	Resolution	Resolution date	Modified	Modified By	Created
	Acetaminophen		Safety First	Init			13-Jun-2014 16:21:51 UTC	Safety First	13-Jun-2014 16:14:19 UTC

From Edit Topic, the user can view attachments for a topic. If the user mouses over the attachment name, a tooltip is displayed with additional information about the attachment.

Figure 2–8 Edit Topic and Attachment table in Empirica Signal

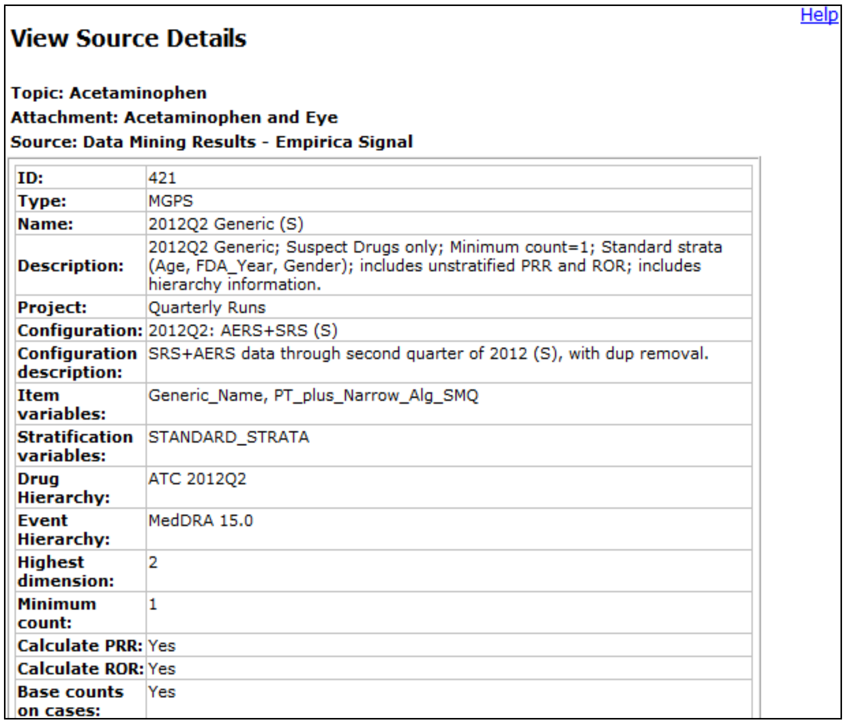

Topic
Preferences Settings Feedback Exit Help
Home Topics
User: Safety First [safety], Topic: Acetaminophen
Print Show All Hide All Back
Topic General Information
Topic Links
Topic Comments
Topic Attachments
Columns Print Download
1 rows
Rows Per Page: 50 Page 1 of 1
Topic Actions
History of Topic

	Attachment name	Attachment description	Source
	Acetaminophen Review		Data Mining Results - Empirica Signal

Run : 2012Q2 Generic (S)
Configuration: 2012Q2: AERS+SRS (S)
# Rows: 448

If the user selects *View Source Details* from the action menu for the attachment, the source details are displayed in a pop-up window. This window includes the topic name, attachment name, and source at the top of the window.

Figure 2–9 View Source Details in Empirica Signal



View Source Details

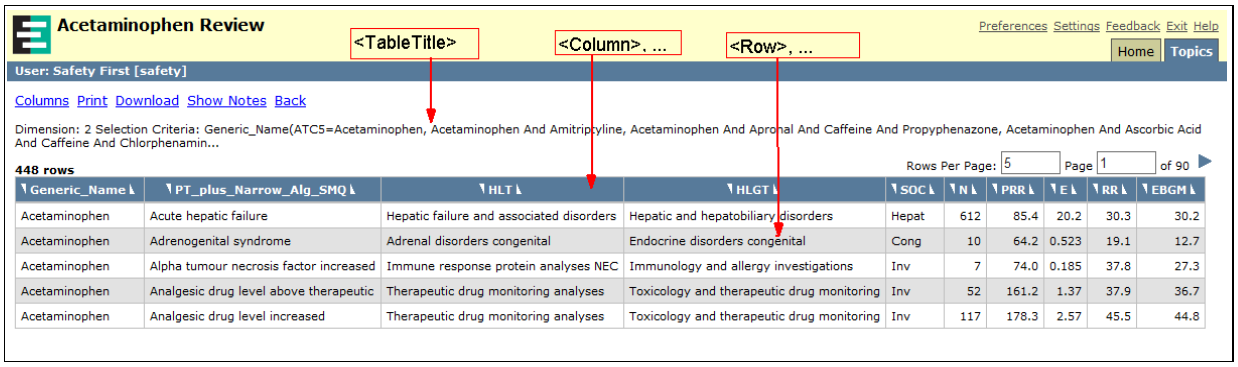
Topic: Acetaminophen
Attachment: Acetaminophen and Eye
Source: Data Mining Results - Empirica Signal

ID:	421
Type:	MGPS
Name:	2012Q2 Generic (S)
Description:	2012Q2 Generic; Suspect Drugs only; Minimum count=1; Standard strata (Age, FDA_Year, Gender); includes unstratified PRR and ROR; includes hierarchy information.
Project:	Quarterly Runs
Configuration:	2012Q2: AERS+SRS (S)
Configuration description:	SRS+AERS data through second quarter of 2012 (S), with dup removal.
Item variables:	Generic_Name, PT_plus_Narrow_Alg_SMQ
Stratification variables:	STANDARD_STRATA
Drug Hierarchy:	ATC 2012Q2
Event Hierarchy:	MedDRA 15.0
Highest dimension:	2
Minimum count:	1
Calculate PRR:	Yes
Calculate ROR:	Yes
Base counts on cases:	Yes

To view the attachment, the user selects View from the action menu in the Topic Attachments table. If the attachment is of type TABLE and there are notes associated with the table, a **Show Notes** link is displayed above the table.

For TABLE type attachments, you send XML with the metadata and data for the table. For more information, see Appendix A, XML attachment tables.

Figure 2–10 Attachment—View


Acetaminophen Review
<TableTitle>
<Column>, ...
<Row>, ...
Preferences Settings Feedback Exit Help
Home
Topics
User: Safety First [safety]
Columns Print Download Show Notes Back

Dimension: 2 Selection Criteria: Generic_Name(ATC5=Acetaminophen, Acetaminophen And Amitriptyline, Acetaminophen And Apronal And Caffeine And Propyphenazone, Acetaminophen And Ascorbic Acid And Caffeine And Chlorphenamin...

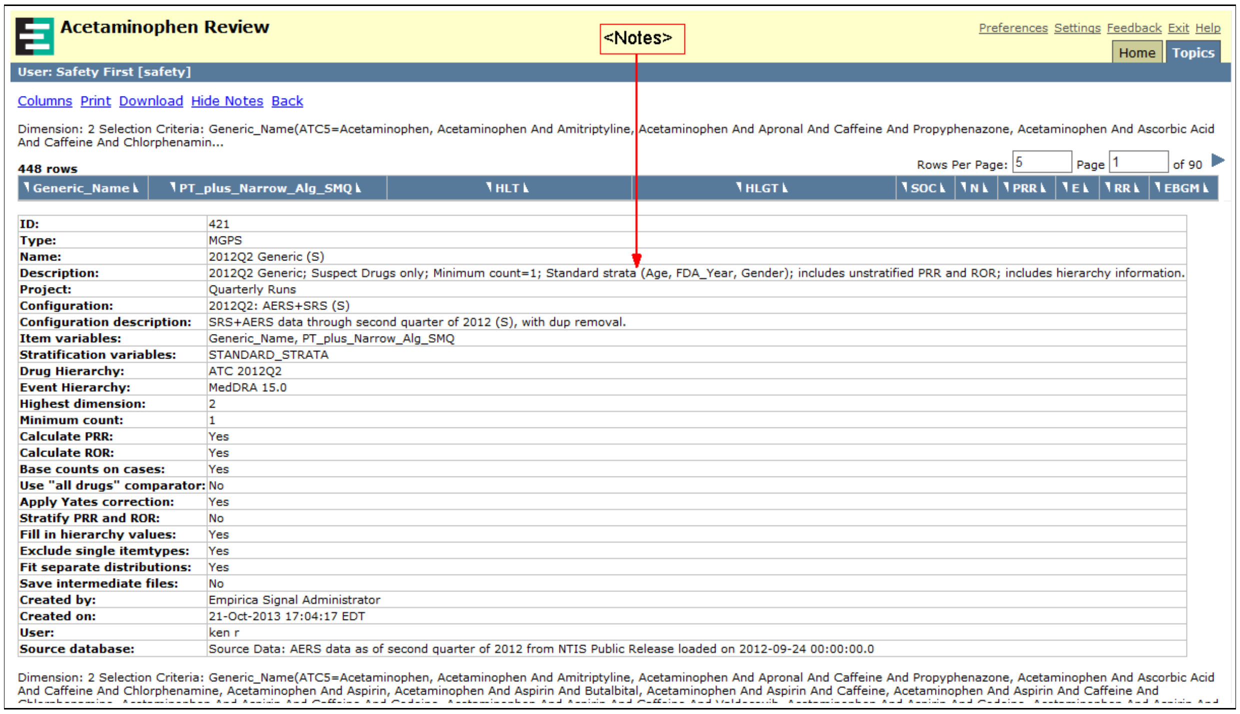
448 rows

Rows Per Page: 5 Page 1 of 90

Generic_Name	PT_plus_Narrow_Alg_SMQ	HLT	HLGT	SOC	N	PRR	E	RR	EBGM
Acetaminophen	Acute hepatic failure	Hepatic failure and associated disorders	Hepatic and hepatobiliary disorders	Hepat	612	85.4	20.2	30.3	30.2
Acetaminophen	Adrenogenital syndrome	Adrenal disorders congenital	Endocrine disorders congenital	Cong	10	64.2	0.523	19.1	12.7
Acetaminophen	Alpha tumour necrosis factor increased	Immune response protein analyses NEC	Immunology and allergy investigations	Inv	7	74.0	0.185	37.8	27.3
Acetaminophen	Analgesic drug level above therapeutic	Therapeutic drug monitoring analyses	Toxicology and therapeutic drug monitoring	Inv	52	161.2	1.37	37.9	36.7
Acetaminophen	Analgesic drug level increased	Therapeutic drug monitoring analyses	Toxicology and therapeutic drug monitoring	Inv	117	178.3	2.57	45.5	44.8

The Figure 2–11 shows the attachment after the **Show Notes** link is selected.

Figure 2–11 Attachment—View and Show Notes



User: Safety First [safety]

Columns Print Download Hide Notes Back

Dimension: 2 Selection Criteria: Generic_Name(ATC5=Acetaminophen, Acetaminophen And Amitriptyline, Acetaminophen And Apronal And Caffeine And Propyphenazone, Acetaminophen And Ascorbic Acid And Caffeine And Chlorphenamin...

448 rows

Rows Per Page: 5 Page 1 of 90

Generic_Name	PT_plus_Narrow_Alg_SMQ	HLT	HLGT	SOC	N	PRR	E	RR	EBGM

ID:	421
Type:	MGPS
Name:	2012Q2 Generic (S)
Description:	2012Q2 Generic; Suspect Drugs only; Minimum count=1; Standard strata (Age, FDA_Year, Gender); includes unstratified PRR and ROR; includes hierarchy information.
Project:	Quarterly Runs
Configuration:	2012Q2: AERS+SRS (S)
Configuration description:	SRS+AERS data through second quarter of 2012 (S), with dup removal.
Item variables:	Generic_Name, PT_plus_Narrow_Alg_SMQ
Stratification variables:	STANDARD_STRATA
Drug Hierarchy:	ATC 2012Q2
Event Hierarchy:	MedDRA 15.0
Highest dimension:	2
Minimum count:	1
Calculate PRR:	Yes
Calculate ROR:	Yes
Base counts on cases:	Yes
Use "all drugs" comparator:	No
Apply Yates correction:	Yes
Stratify PRR and ROR:	No
Fill in hierarchy values:	Yes
Exclude single itemtypes:	Yes
Fit separate distributions:	Yes
Save intermediate files:	No
Created by:	Empirica Signal Administrator
Created on:	21-Oct-2013 17:04:17 EDT
User:	ken r
Source database:	Source Data: AERS data as of second quarter of 2012 from NTIS Public Release loaded on 2012-09-24 00:00:00.0

Dimension: 2 Selection Criteria: Generic_Name(ATC5=Acetaminophen, Acetaminophen And Amitriptyline, Acetaminophen And Apronal And Caffeine And Propyphenazone, Acetaminophen And Ascorbic Acid And Caffeine And Chlorphenamine, Acetaminophen And Aspirin, Acetaminophen And Aspirin And Butalbital, Acetaminophen And Aspirin And Caffeine, Acetaminophen And Aspirin And Caffeine And

3 API

This chapter includes the following sections that describe each topic service method call in detail:

API summary

The API includes the following topic service methods 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.

Table 3–1 Topics Service API summary.

Operation	Method Name	Description
Get a list of topics	*getTopicContexts*	Retrieves topic names and IDs.
Get a list of actions	*getActionContexts*	Retrieves action names and IDs for a topic.
Get a list of work teams	*getWorkteamContexts*	Retrieves work team names and IDs.
Get a list of topic templates	*getTopicTemplateContexts*	Retrieves topic template names and IDs.
Attach to an existing topic	*attachTopic*	Creates an attachment to an existing topic.
Attach to a new topic	*attachNewTopic*	Creates an attachment to a new topic.
Attach to an action	*attachAction*	Creates an attachment to an existing action.
Get field metadata	*getFieldMetadata*	Retrieves field properties for topics, topic templates, or work teams.
Get field values	*getFieldValues*	Retrieves field values for topics, topic templates, or work teams.
Get filters	*getFilters*	Retrieves fields defined as filters and their values for topics, topic templates, or work teams.
Get a list of projects	*getProjects*	Retrieves projects used by the user's topics.
Get user info	*getUserInfo*	Retrieves topic permissions for the user such as, can the user save to a topic.
Get topics service properties	*getTopicsServiceProperties*	Retrieves the topics service version and other topics service properties. This can be used to determine whether the topics service is available and is (or has) a version compatible with your version of the API.

Common value objects

Topics service methods contain value objects that are passed as arguments or returned from service calls. This document describes the member variables for each value object. If you are importing the WSDL with Java, the value objects have protected member variables and your application will access the variables through public getters and setters that are generated from the WSDL. These fields are available in the *getFieldName* and *setFieldName* form. If you are importing the WSDL with .NET, the member variables are public rather than having getters and setters.

TopicsServiceContext value object

Most methods have a *TopicsServiceContext* value object passed as a parameter. This value object contains context information to process a request. The *username* field is required. You can also set optional parameters.

startRow and ***chunkSize***—To page the resulting data by making multiple calls to the method with a starting row number and the number of rows to return with each call.

Table 3–2 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 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 Topics API that you are using. Empirica Signal Release 8.0 contains version 2 of the Empirica Topics API as described in this document. If your Topics API version is greater than the 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* method and is ignored for any other methods. If this field is not specified, a default sort order is used.
filter	Input	List of columns on which to filter data and the filter value to match on. This field applies to only the *getFieldValues* method and is ignored for any other methods. If this field is not specified, the data is returned without any filtering.
startRow	Input	Start row from which to begin retrieving data. If this field is not specified, the default value is 1.
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* methods. These fields are ignored for other methods.
nameContainsFilter	Input	Case-insensitive wildcard filter for topic names. This field applies only to the *getTopicContexts* method and is ignored for any other methods. If specified, this field filters the results to topic names that contain this text.
version	Output	Topics service version of the Empirica Topics server.

Table 3–2 (Cont.) TopicsServiceContext optional parameters.

Field name	Parameter type	Description
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 Empirica Topics server. You can also get this property from the *getTopicsServiceProperties* method.
numRows	Output	Number of rows that were retrieved by the method. This field is less than or equal to the chunk size.
totalRows	Output	Total number of rows available for the method. 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*, and *getFieldMetadata*, *getFieldValues* methods.

```
class TopicsServiceContext {
  String applicationName;
  String applicationInstance;
  Integer version;
  String username;
  Integer maxRows;
  Integer startRow;
  Integer chunkSize;
  Integer numRows;
  Integer totalRows;
  List<TopicsSortOrder> sortOrder;
  List<TopicsFilter> filter;
  String nameContainsFilter;
}
```

Methods

This section discusses the Empirica Topics methods in detail.

Get a list of topics

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

```
TopicContexts getTopicContexts(TopicsServiceContext serviceContext,
                               long findTopicId)
throws InvalidArgumentException, TopicsServiceException;
```

This method 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 method 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 **Imp**ortant Results and Topics **imp**orted.

TopicsContexts value object

The *getTopicContexts* method returns TopicContexts, which has a list of topic IDs and names. 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.

Note: Topic names are not unique.

```
class TopicContexts extends TopicsServiceResult {
  TopicContext matchingTopicContext;
  List<TopicContext> topicContexts;
}

class TopicsServiceResult {
  TopicsServiceContext topicsServiceContext;
}

class TopicContext {
  long id;
  String name;
}
```

Get a list of actions

Your application retrieves a list of actions for a topic available to the user by calling the *getActionContexts* method, which 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 Empirica Signal application adds the options **--none--** to the top of the drop-down list. Optionally, you can set a start row and a chunk size.

```
ActionContexts getActionContexts(TopicsServiceContext topicsServiceContext,
                                 long topicId)
throws InvalidArgumentException, TopicsServiceException;
```

ActionContexts value object

The value object of the *getActionContexts* method returns a list of ActionContexts with action IDs and names and the associated topic ID.

```
class ActionContexts extends TopicsServiceResult {
  List<ActionContext> actionContexts;
}

class ActionContext {
  long id;
  String name;
  long topicId;
}
```

Get a list of work teams

Your application can retrieve a list of work teams by using the *getWorkteamContexts* method. This method 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 using Save to Topic with a Topic Workflow Configuration that is set to **one and only one work team**, this method can be used to populate the list of work teams in the **Save to Topic** dialog box.

```
WorkteamContexts getWorkteamContexts (TopicsServiceContext serviceContext)
throws InvalidArgumentException, TopicsServiceException;
```

WorkteamContexts value object

The *getWorkteamContexts* method returns *WorkteamContexts* with a list of work team names and IDs.

```
class WorkteamContexts extends TopicsServiceResult {
  List<WorkteamContext> workteamContexts;
}

class WorkteamContext {
  long id;
  String name;
}
```

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* method, which returns a list of topic template contexts with the name and identifier of each topic template. This method 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 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.

```
TopicTemplateContexts getTopicTemplateContexts(TopicsServiceContext
serviceContext)
throws InvalidArgumentException, TopicsServiceException;
```

TopicTemplateContexts value object

The *getTopicTemplateContexts* method returns *TopicTemplateContexts* with a list of topic template IDs and names.

Note: Topic template names are not unique.

```
class TopicTemplateContexts extends TopicsServiceResult {
  List<TopicTemplateContext> topicTemplateContexts;
}

class TopicTemplateContext extends BaseTopicContext {}
class BaseTopicContext {
 long id;
 String name;
}
```

Attach to an existing topic

Your application can create an attachment to an existing topic. The topic argument in the *attachTopic* method contains the identifier of the topic. Set the *id* field of the *topicContext* argument to the ID of the topic to attach to. The *attachmentInput* argument has the name, extension, attachment type, and optional properties of the attachment. The *dataHandler* field is a W3C MTOM (Message Transmission Optimization Mechanism) type to stream data.

The attachment types are:

- **IMAGE**—A ZIP file with a series of JPG, JPEG, PNG, or GIF images, optional *.txt files, and optional `notes.txt` files. 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. For example, a case series table. See Appendix A, XML attachment tables for the XML definition.

Typically, your application writes an IMAGE or TABLE attachment to a temporary file, streams the file using the *attachTopic*, *attachNewTopic*, or *attachAction* method, and then deletes the file. A .NET client calls the *streamFileAttachment* method first, and then calls the *attachTopic*, *attachNewTopic*, or *attachAction* method.

```
void attachTopic(TopicsServiceContext topicsServiceContext,
                 TopicContext topicContext, AttachmentInput attachmentInput)
throws InvalidArgumentException, TopicsServiceException
```

AttachmentInput value object

The attachment input value object includes the following fields:

- ***name***—Required name field.
- ***description***—Optional description.
- ***source***—Required brief name of the source, such as “Cases”.
- ***sourceText***—Required brief information about the attachment. 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. The text is limited to 500 characters by Internet Explorer.

  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 of *IMAGE*, *FILE*, or *TABLE*. The types *NOTE*, *URL*, and *SIGNAL* are reserved for future use.
- ***extension***—Name of the extension. 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 JPG,

JPEG, PNG, or GIF images, optional *.txt files with strings such as section headers, and optional `notes*.txt` files.

- ***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.

- ***dataHandler***—Used by Java clients to stream data. The DataHandler is a MTOM (W3C Message Transmission Optimization Mechanism) for streaming data to transfer large amounts of data. It can be used for large or binary files.

  For .NET, the *dataHandler* variable is not used and you should call the *streamFileAttachment* method to stream your attachment.

- ***guid***—Used by .NET clients when streaming data. The *guid* uniquely identifies the attachment across two method calls, the first to stream the attachment and then the second to attach to a topic or action.

- ***urlAddress***—Reserved for future use.

- ***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.

```
class AttachmentInput {
  String name;
  String description;
  String source;
  String sourceText;
  AttachmentType type;
  String extension;
  String notes;
  String data;
  DataHandler dataHandler;
  String guid;
  String urlAddress;
}

enum  AttachmentType {
  NOTE, IMAGE, FILE, URL, SIGNAL, TABLE
}
```

Attach to a new topic

Your application can create an attachment to a new topic. The *attachNewTopic* method creates an attachment to a new topic. For information about the *attachmentInput* argument, see *Attach to an existing topic*.

attachNewTopic method

```
TopicResult attachNewTopic(TopicsServiceContext topicsServiceContext,
                           TopicInput topicInput, AttachmentInput attachmentInput)
throws InvalidArgumentException, TopicsServiceException;
```

TopicInput value object

The *TopicInput* value object has the required name field for the topic name and the following optional parameters:

- *projectName*—Name to categorize the topic (required if newProject is true)
- *description*
- *workteams*—A list of work team names
- *newProject*—A boolean set to true to create new project
- *templateId*—ID for the topic template

Any optional values must be set to null if they are not defined otherwise.

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 the user does not select a topic template, set the template ID to null.

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

```
class TopicInput {
  String name;
  String projectName;
  String description;
  String[] workteams;
  boolean newProject;
  Long templateId;
}
```

Attach to an action

Your application can create an attachment to an existing action by calling the *attachAction* method. 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 *attachmentInput* argument, see *Attach to an existing topic*.

```
void attachAction(TopicsServiceContext topicsServiceContext,
                  ActionContext  actionContext, AttachmentInput attachmentInput)
throws InvalidArgumentException, TopicsServiceException;
```

Get field metadata

Your application can retrieve a list of all fields for topics, topic templates, or work teams by calling the *getFieldMetadata* method. This method 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. Set the *fieldContext* argument to one of the enumeration values of FieldContext: *TOPICS*, *TOPIC_ TEMPLATES*, or *WORKTEAMS*. This returns the *FieldMetadata* value object with a list of field properties.

```
FieldMetadata getFieldMetadata(TopicsServiceContext topicsServiceContext,
                               FieldContext fieldContext)
throws InvalidArgumentException, TopicsServiceException;
```

The *FieldContext* input parameter is an enumeration for topics, topic templates, or work team values.

```
enum FieldContext {
  TOPICS, TOPIC_TEMPLATES, WORKTEAMS;
}
```

FieldMetadata value object

The *FieldMetadata* value object has a list of metadata of all fields in each topic, topic template, or work team. The metadata 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.

```
class FieldMetadata extends TopicsServiceResult {
  List<Field> fields;
}

class Field {
  String id;
  String label;
  FieldType type;
  boolean filterable;
  boolean required;
}
```

The field types are defined in an enumeration as follows. *FLOAT* is reserved for future use.

```
enum FieldType {
  STRING, DATE, INTEGER, LONG, FLOAT, PROJECT, STATE
}
```

Get field values

Your application can retrieve a list of all field values for each topic, topic template, or work team by calling the *getFieldValues method*. This method 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*, or *WORKTEAMS*. This method 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.

```
FieldValues getFieldValues(TopicsServiceContext serviceContext,
                           FieldContext fieldContext)
throws InvalidArgumentException, TopicsServiceException;
```

The *FieldContext* input parameter in an enumeration for topics, topic templates, or work team values.

```
enum FieldContext {
  TOPICS, TOPIC_TEMPLATES, WORKTEAMS
}
```

FieldValues value object

The *FieldValues* value class has an *ids* field, which is an ordered list of identifiers, and a *values* field, which is a list of arrays of field values associated with the ids.

```
class FieldValues extends TopicsServiceResult {
  ArrayList<Long> ids;
  ArrayList<FieldValueArrayGenerated> values;
}

class FieldValueArrayGenerated {
  FieldValue[] values;
}
```

Each *FieldValue* object has the field *type*, the name of the field, and a field value depending on the type. *STRING* types are returned as *stringValue* fields. *INTEGER* or *LONG* types are returned as a *longValue* fields. *DATE* types are returned as *dateValue* fields. The *nameIdPair* field is used when the *type* is *PROJECT* or *STATE* and returns pairs of the identifier and name for the project or state.

The *floatValue* field is reserved for future use.

```
class FieldValue {
  FieldType type;
  String fieldName;
  String stringValue;
  Long longValue;
  Date dateValue;
  NameIdPair nameIdPair;
  Float floatValue;
}

class NameIdPair {
  String id;
  String name;
}
```

Get filters

Your application can retrieve a list of fields that have been set up as filters and their values by calling the *getFilter*s method.

```
Filters getFilters(TopicsServiceContext topicsServiceContext,
                   FieldContext fieldContext)
throws InvalidArgumentException, TopicsServiceException;
```

Filters value object

The Filters object is returned from the *getFilters* method and is an ordered list of filter field names. You can 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.

```
class Filters extends TopicsServiceResult {
  ArrayList<String> fieldNames;
  ArrayList<FieldValuePairGenerated> values;
}

class FieldValuePairGenerated {
  FieldValuePair[] values;
}

class FieldValuePair {
  String label;
  FieldValue value;
}
```

Filtering results

An end user might use this method, for example, when looking for topics with a drug name in a custom field.

```
class TopicsServiceContext {
  …
  List<TopicsSortOrder> sortOrder;
  List<TopicsFilter> filter;
  …
}
```

After the end user selects one or more filters, set the *filter* field in the *TopicsServiceContext* when calling *getTopicContexts*, *getFieldValues, and getProjects*. *TopicsFilter* is a list of filter field names and the selected filter values. 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.

Sorting results

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

```
class TopicsServiceContext {
  …
  List<TopicsSortOrder> sortOrder;
  List<TopicsFilter> filter;
  …
}
```

TopicsSortOrder value object

Set the *fieldName* field to the name of the topic, topic template, or work team 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 method applies to only the *getFieldValues* method.

```
class TopicsSortOrder {
  String fieldName;
  FieldType fieldType = FieldType.STRING;
  Order order = Order.SORT_ASC;
}
```

The field types are defined in an enumeration as follows:

```
enum FieldType {
  STRING, DATE, INTEGER, LONG, FLOAT, PROJECT, STATE
}

enum Order {
  SORT_ASC, SORT_DESC
}
```

Get a list of projects

Your application can retrieve a list of existing projects by calling the *getProjects* method. A project can be optionally specified when creating a topic. This method returns the *ProjectList* value object that always includes a value called **Unassigned** with ID **0**.

```
ProjectList getProjects(TopicsServiceContext serviceContext)
throws TopicsServiceException
```

ProjectList value object

```
class ProjectList extends TopicsServiceResult {
  List<String> projects;
}
```

Get user info

Your application can view the topic permissions for a user by calling the *getUserInfo* method. Your application might hide or disable the **Save to Topic** link if the user cannot save to Topics.

```
TopicsUserInfo getUserInfo(TopicsServiceContext topicsServiceContext,
                           String username)
throws InvalidArgumentException, TopicsServiceException;
```

TopicsUserInfo value object

The *TopicsUserInfo* method 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.

```
class TopicsUserInfo extends TopicsServiceResult {
  String username;
  boolean canSaveToTopics;
  boolean canSaveToNewTopic;
  boolean canAccessMultipleWorkteams;
  boolean canViewTopics;
}
```

Get topics service properties

Your application can retrieve the Topics server version and other properties by calling the *getTopicsServiceProperties* method.

```
TopicsServiceProperties getTopicsServiceProperties();
```

TopicsServiceProperties value object

The *getTopicsServiceProperties* method returns the *TopicsServiceProperties* value object with the following fields:

- ***serverVersion***—The version of the topics server. If your API version is greater than the Topics server version, you might experience compatibility issues.

  Note: Topics services are backward compatible with earlier client versions.

- ***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.

```
class TopicsServiceProperties {
  Integer serverVersion;
  Integer attachMaxRows;
  Integer attachMaxMegabytes;
}
```

Exceptions

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

Topics service methods 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 an *errorCode* field that categorizes the exception, a *message* field with a default message that can be displayed to the user, and a *detailedMessage* field for logging and diagnostics.

You can either display the default message or create a customized message for a particular error code. For example, if there is a missing topic name for a new topic, the customized message could be **The new topic name is required** instead of the default message **You must enter a new topic name.**

TopicsServiceException value object

```
class TopicsServiceException extends Exception {
  String detailedMessage;
  TopicsServiceErrorCode errorCode;
  String message;
}
```

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* method called with the *topicInput* parameter *name* field not set.
TOPIC_ERROR_ATTACHMENT_NAME_ REQUIRED	*attachTopic, attachNewTopic,* or *attachAction* method called with the *attachmentInput* parameter *name* field not set.
TOPIC_ERROR_PROJECT_NAME_REQUIRED	*attachNewTopic* method called with *topicInput parameternewProject* field set but *projectName* field not set.
TOPIC_ERROR_TOPIC_CLOSED	*attachTopic* method called with the *topicContext* parameter field *id* set to a closed topic.
TOPIC_ERROR_ACTION_CLOSED	*attachAction* method called with the action*Context* parameter field *id* set to a closed action.
TOPIC_ERROR_TOPIC_DELETED	*attachTopic* method called with the *topicContext* parameter field *id* set to a deleted topic.
TOPIC_ERROR_ACTION_DELETED	*attachAction* method called with the action*Context* parameter field *id* set to a deleted action.
TOPIC_ERROR_UNAVAILABLE_TOPIC	*attachTopic* method called with the *topicContext* parameter field *id* set to a topic that is not available.
TOPIC_ERROR_UNKNOWN_ACTION	*attachAction* method called with the action*Context* parameter field *id* set to an action that is not available.
TOPIC_ERROR_UNKNOWN_TOPIC	Reserved for future use.

```
enum TopicsServiceErrorCode {
  TOPIC_ERROR_INTERNAL,
  TOPIC_ERROR_NO_SERVICE,
  TOPIC_ERROR_TOPIC_NAME_REQUIRED,
  TOPIC_ERROR_ATTACHMENT_NAME_REQUIRED,
  TOPIC_ERROR_PROJECT_NAME_REQUIRED,
  TOPIC_ERROR_TOPIC_CLOSED,
  TOPIC_ERROR_ACTION_CLOSED,
  TOPIC_ERROR_TOPIC_DELETED,
  TOPIC_ERROR_ACTION_DELETED,
  TOPIC_ERROR_UNAVAILABLE_TOPIC,
  TOPIC_ERROR_UNKNOWN_TOPIC
}
```

Upload files for .NET

.NET clients must first stream files to the server and include a GUID in the attachmentInput parameter, and then call one of the attach methods *attachTopic, attachNewTopic,* or *attachAction*. Java clients do not use this method as the Java *dataHandler* field in *AttachmentInput* class handles the streaming of the file.

```
void streamFileAttachment(TopicsServiceContext topicsServiceContext, String guid,
                          byte[] dataHandler, int bufferSize, int offSet)
throws InvalidArgumentException, TopicsServiceException;
```

When importing the WSDL with.NET, the *dataHandler* parameter is a byte array.

A
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

```
<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 above XML code illustrates the format for the XML that is streamed for TABLE attachments. 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 2–10 and Figure 2–11. 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.

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

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

c. `<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.

d. `<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**.

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 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.
- **Table**—An XML representation of a table. See Appendix A, XML attachment tables for more information.

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.

topics service

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

topics service context

A parameter to every topics method that includes context information such as the username and information for sorting, filtering, and paging.

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.

username

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.

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.

work team

A work team is a subset of users from an 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.