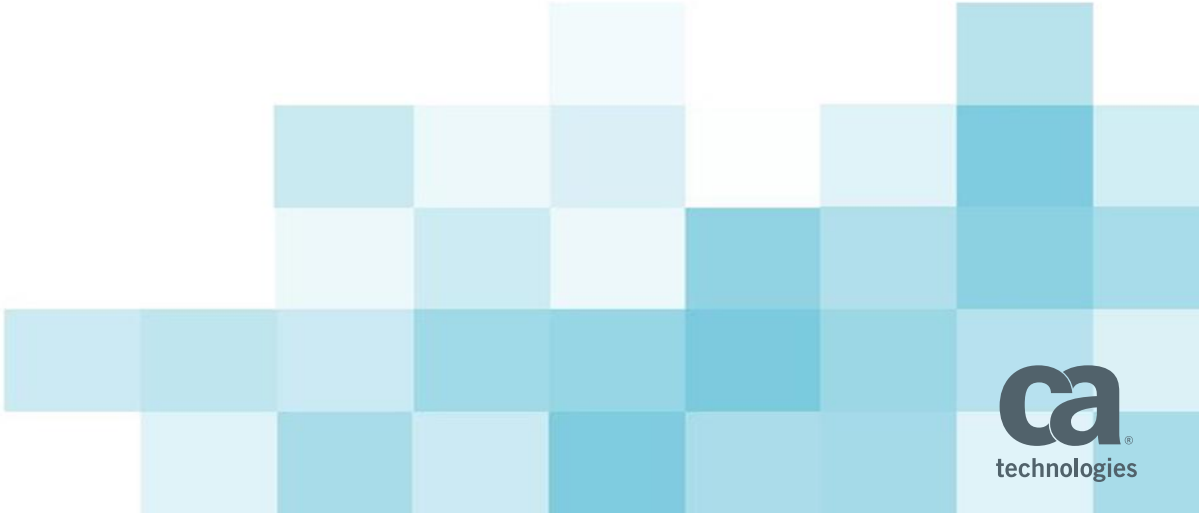
CA Spectrum: Working with REST API's

30th August 2016

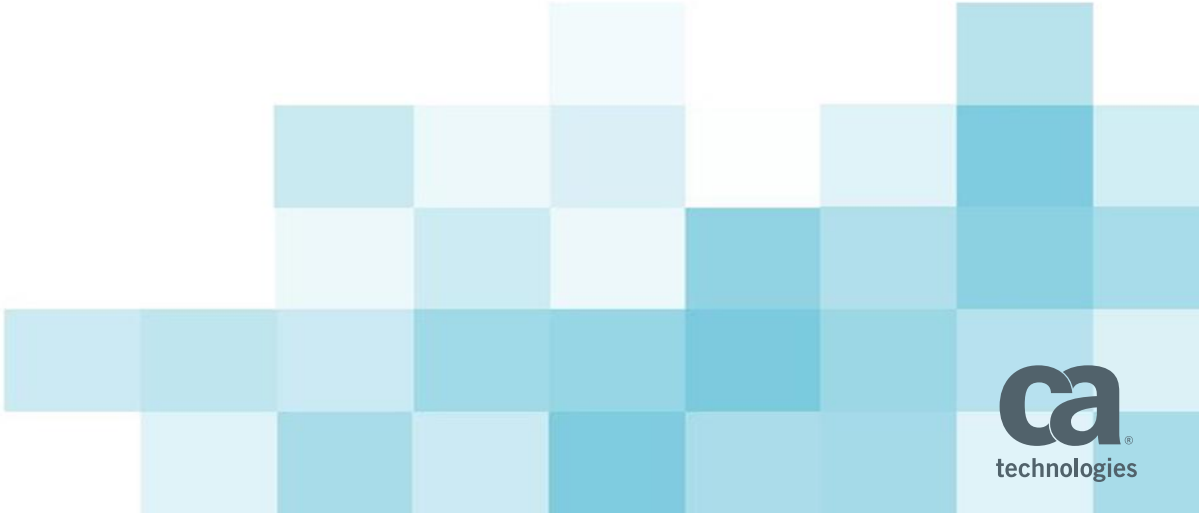


ca technologies

Agenda

1. What is a REST Webservice?
2. Spectrum functions through Restful Web Services.
3. Spectrum Web services Architecture.
4. Working with Spectrum Web Services API.
5. Q&A

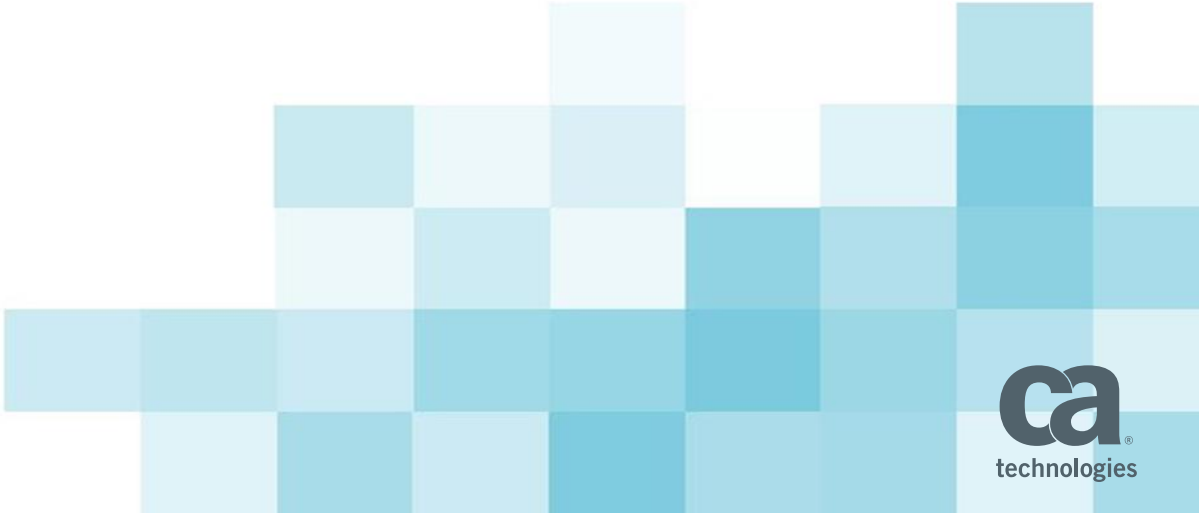
REST WebServices



REST WebServices

- Representational State Transfer (**REST**)
- The REST architecture is a lightweight **HTTP/HTTPS** -based approach for SOAPless Web Services.
- Supported Operations: **POST** (create), **GET** (read), **PUT** (update), **DELETE**, HEAD, OPTIONS, and TRACE.
- RESTful architecture and applications are **stateless**, which means that no client context information is stored between requests. Each request contains all the information necessary to service the request.

CA Spectrum WebServices



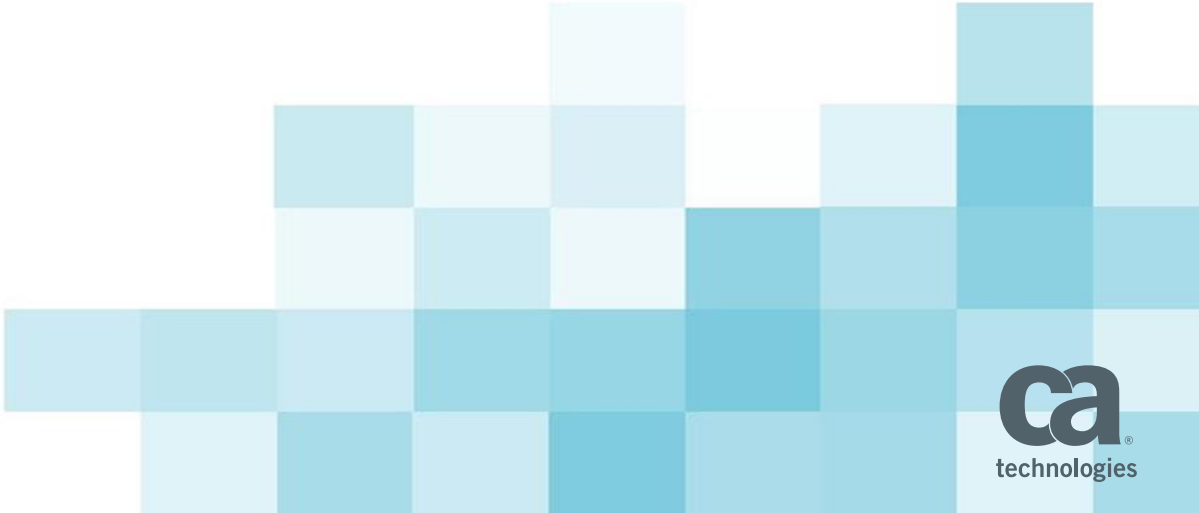
Spectrum supports RESTful Web Services

- The CA Spectrum Web Services API supports the REST architecture.
- Using the CA Spectrum Web Services API, CA Spectrum data can be accessed directly from a browser or integrated into your own application.

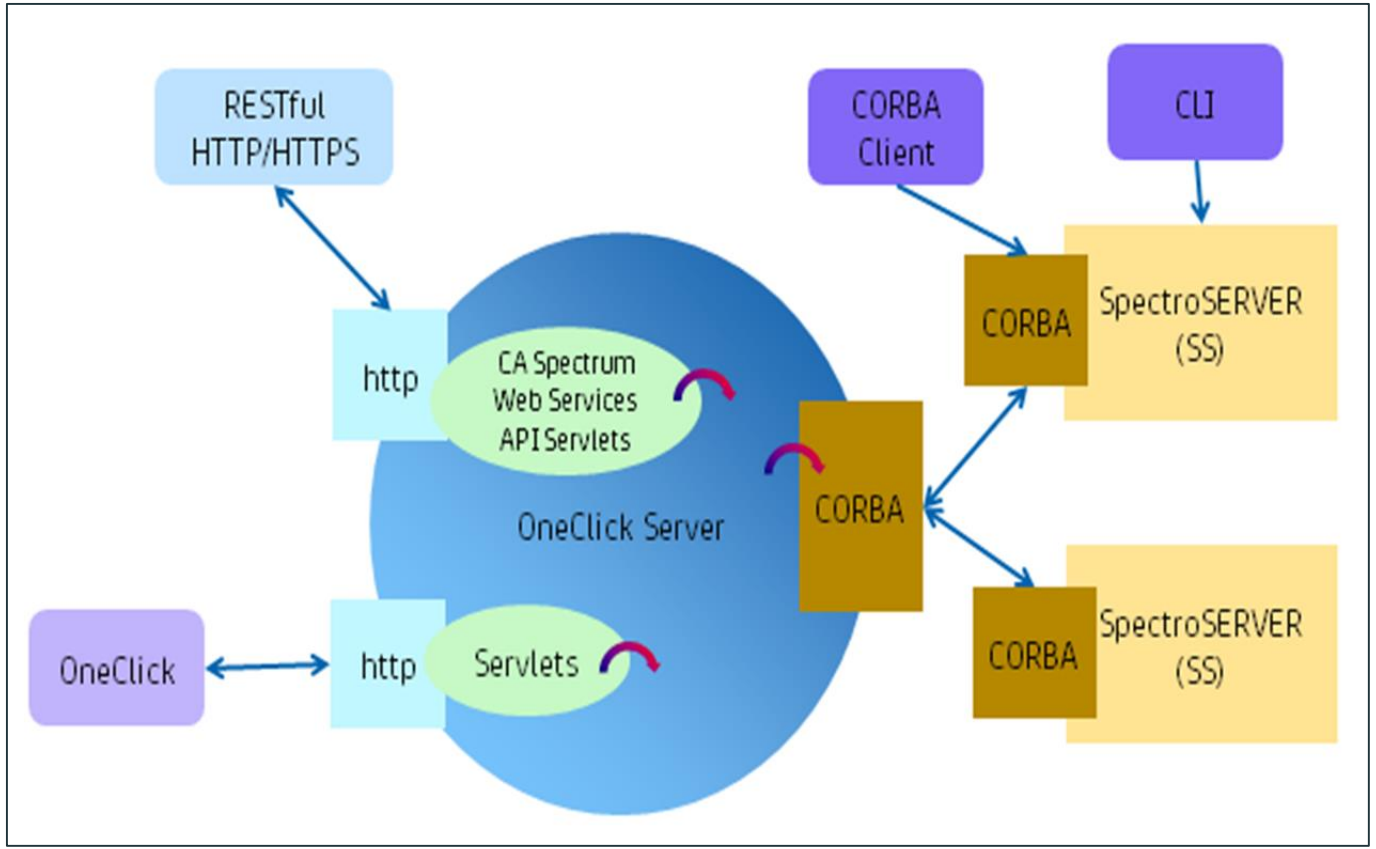
Spectrum Functions using Restful

- By using the CA Spectrum Web Services API, you can take advantage of functionality provided by the OneClick server, such as:
  - Access devices, models, relationships, attributes, actions and alarms.
  - Manage devices, ports, containers, services, and links.
  - Read, update, and clear alarms.
  - Manage subscriptions and notifications.
  - Create Device models and Global Collections etc..

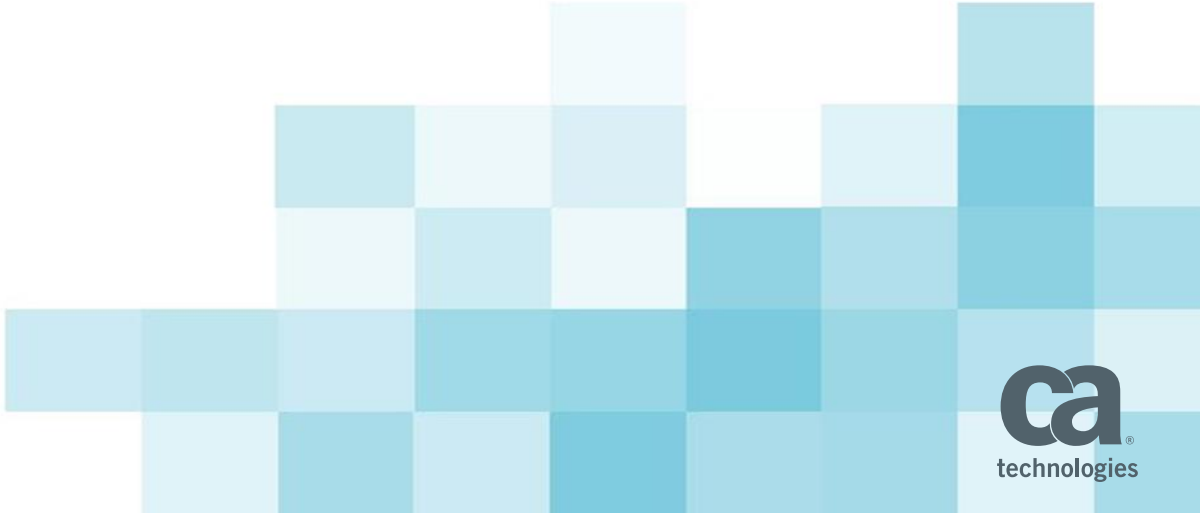
CA Spectrum WebServices Architecture



Spectrum Web services Architecture



Working with Spectrum WebServices API



Supported REST Functions

Spectrum Web Services supports only these 4 REST operations:

- **POST** (Create)
- **GET** (Read)
- **PUT** (Update)
- **DELETE** (Delete)

Note: The HEAD, OPTIONS, and TRACE functions are not supported in the CA Spectrum Web Services API.

Working with Spectrum Web Services API

- Request Syntax:

http://<*hostname*><:*portnumber*>/spectrum/restful/***<request>***

Here **<request>** is spectrum's Restful **resource**

For e.g., a <request> can be devices, alarms, landscapes, model etc.

Use Case - DEMO

- MODEL LIFECYCLE
  - Create a Model
  - Retrieve the created Model
  - Retrieve an attribute of the created Model.
  - Read the alarms on the Model.
  - Subscribe for Alarms on the Model.
  - Delete the Model.

Create Model using POST Operation

- Syntax To create a new Model:

*http://<hostname><:portnumber>/spectrum/restful/**model**[?landscapeid=<landscape_handle>] [&commstring=<comm_str>] [&ipaddress=<ip_address>][&parentmh=<model_handle>]*

- Creates a new device model and returns model_handle.

Read Model using GET Operation

- Use **model** as Resource.

To retrieve attributes from the specified model.

*http://<hostname><:portnumber>/spectrum/restful/**model**/<model_handle>[?attr=<attr_ID>]*

Devices using GET Operation

- Use **devices** as Resource

To retrieve all device model handles:

http://<hostname><:portnumber>/spectrum/restful/devices

Result: Returns model-handles of all available devices.

- To get specific device attributes:

http://<hostname><:portnumber>/spectrum/restful/devices[?attr=<attr_ID>][&landscape=<landscape_handle>][&throttlesize=<num>]

Alarms using GET operation

Use **alarms** as Resource

- To retrieve all the alarms from SS:

http://<hostname><:portnumber>/spectrum/restful/alarms

Result: Returns Alarm-Id's of all the alarms.

- Retrieve specific alarms:

http://<hostname><:portnumber>/spectrum/restful/alarms[?attr=<attr_ID>][&landscape=<landscape_handle>][&throttlesize=<num>]

Alarms using DELETE operation

To Delete an Alarm:

http://<hostname><:portnumber>/spectrum/restful/alarms/<alarm_id>

The alarm represented by the alarm_id will be deleted.

Model using DELETE Operation

To Delete an existing model:

http://<hostname><:portnumber>/spectrum/restful/model/<model_handle>

SUBSCRIPTION

Use the ***Subscription*** resource to create or retrieve subscriptions. A subscription is a request to be notified of activity on any of the following:

- Models/attributes. Registers watches on specified models and related attribute changes.
- Alarms/attributes. Registers watches for alarm creation/clearing and attribute changes.

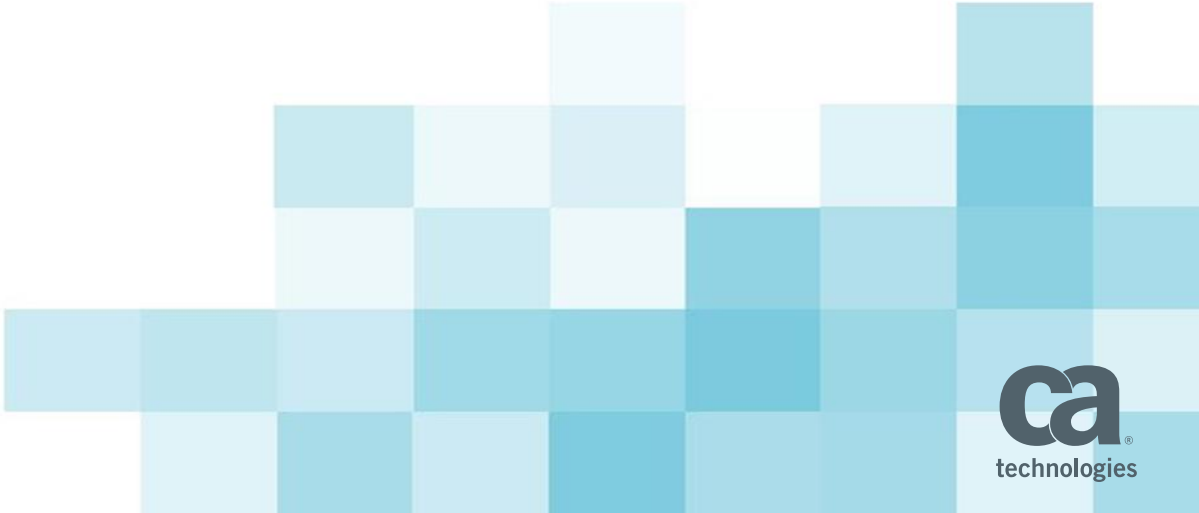
Base URL :
http://<hostname><:portnumber>/spectrum/restful/subscription

Types of Subscription: PULL & PUSH

- Subscriptions are *pull* or *push*. A pull subscription requires that the client poll the subscription ID every time to retrieve notification if any, whereas a push subscription requires that the client provide a URL to which notifications can be POSTed.
- Notifications contain change information in XML or JSON format.
- Currently subscriptions will be pushed to Tomcat logs and it only supports XML format.

Demo:

SUBSCRIPTION – NOTIFICATION



ca technologies

Questions?