



Predix Developer Bootcamp

October 2016

Confidential. Not to be copied, distributed, or reproduced without prior approval.

Welcome to the Predix Developer Bootcamp course. This comprehensive, multi-day course is designed to provide developers with a foundation for working with the Predix platform services.



Course Overview

Description	This course provides participants with the knowledge and basic skills necessary to build industrial applications using Predix
Audience	This course is intended for Developers, and those who build, deploy and support Predix solutions
Objectives	<p>Upon completion of this course, you will be able to:</p> <ul style="list-style-type: none">• Deploy and manage services in Predix• Secure applications using Predix Security services• Ingest, manage and model data using Predix Data services• Execute, orchestrate and catalog analytics• Generate a Predix machine and configure data flow

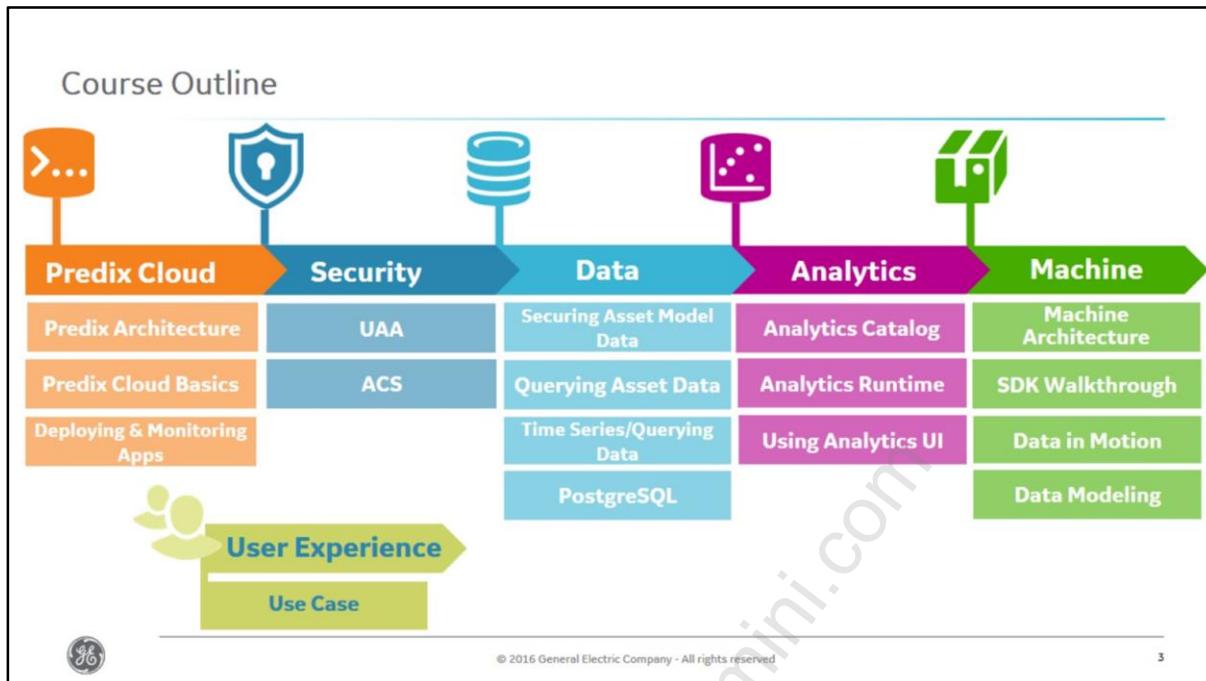


© 2016 General Electric Company - All rights reserved.

2

Here is a brief look at the intended audience and objectives of the course.





- This is a 5-day class that will provide participants with key concepts and hands-on lab exercises.
- It is designed to help developers progressively apply the concepts as they are being taught.
- Hands-on lab activities further engrain the concepts presented.
- In the 'Course Outline' diagram, each color represents a module.



Welcome

- Name
- Business Unit
- Job Role
- Expectations
- Fun Fact



© 2016 General Electric Company - All rights reserved.

4

Before we get started, your instructor will cover a few "housekeeping" details.

- Class schedule – class runs from 9:00am – 5:00pm Monday through Thursday. Class ends at 2:00pm on Friday (class schedule may fluctuate to accommodate locations and students' needs).
- Lunch – There is a 1-hour lunch break each day. Your instructor will point out eating establishments in your area.
- Restrooms – your instructor will point out locations for restrooms for your training site.
- Please have phones on silent mode; if you must take a phone call, please step outside the classroom.



Lab Environment & Course Materials



Lab Environment: Fast Lane

Your account will give you access to the lab environment until the end of the class:

<https://gedpdx.remotelabs.io>

Course Materials: Content Raven

You will receive an email containing a link to the site where you can access the course materials.

New content has been securely sent to you.
Check the file names on the right to see what has been shared with you.

Simply click on the individual file names or Click to view to see all of your content.

Thank you,
Vicki

Shared Files:

Click link below to launch content.



Predix Developer Boot Camp

[Login to your account](#)



© 2016 General Electric Company - All rights reserved.

5

Course materials – prior to class, students will have received an email with a link to the Content Raven site where they can access the Lab Guide and PowerPoint presentation. You have a one-time option to print the materials; your instructor can direct you to an available printer.

Hosted environment – all lab exercises are completed in a hosted environment (DevBox) managed by a third-party provider (Fast Lane). Prior to class, students will have received an email from Fast Lane with a set of instructions for logging onto the lab environment. The URL to the hosted site is: <https://gedpdx.remotelabs.io>. Your instructor will provide you with an access code in class.



Predix Cloud

Predix Architecture



© 2016 General Electric Company - All rights reserved.

6

This first module introduces the Predix platform, its architecture and features.



Module Overview: Predix Cloud



Here is a list of topics to be covered in this module.



Module Objectives: Predix Cloud

By the end of this module you will be able to:

- Explain the Predix Architecture
- Use Predix Cloud
- Deploy & monitor Predix applications
- Identify Predix Services

Completion of the following lab activities will reinforce the concepts covered in this module:

- Lab 1: Getting Started with Predix Cloud
- Lab 2: Deploying and Monitoring Applications
- Lab 3: Building Microservices



© 2016 General Electric Company - All rights reserved.

8



What is Predix?

Predix is a **software platform** optimized for building and running **industrial internet** applications.

Predix is delivered as a **Platform-as-a-Service** (PaaS) that help companies:

- Capture and analyze massive amounts of complex industrial data
- Create a *system-wide* view of assets – from edge to cloud
- Meet the needs of industrial-grade, cyber and operational security requirements
- Innovate faster, quickly develop and deploy industrial apps
- Leverage an ecosystem of services built and maintained by GE and its partners



© 2016 General Electric Company - All rights reserved.

9

GE Digital offers a complete portfolio of products, solutions and services that help leading industrial enterprises drive digital transformation. At the heart of this portfolio is **Predix** - the platform for the Industrial Internet.

Purpose-built for industry, it empowers organizations to develop, deploy, and operate industrial apps—driving outcomes such as **reduced unplanned downtime, improved asset output, and greater operational efficiency**. Decades of experience in industries from power generation to manufacturing to healthcare have enabled GE to create a platform that meets the unique needs of industry.

Predix is a cloud-based platform that runs in a cloud managed by GE. By leveraging a core set of services and infrastructure provided by the platform, companies can build a *system-wide* view of their assets. This view allows both improved optimization of each part in the system as well as optimization of the entire system. This is the unique “edge-to-cloud” coverage offered by the Predix platform.

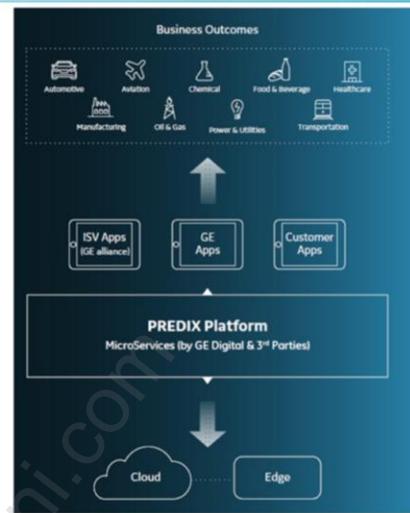
As a leader in the industrial world, GE built a cloud that meets the needs industrial companies have for scale, security, and regulatory compliance. Predix cloud can handle vast amounts of Industrial Internet information, while also managing customer SLAs, security, support, governance, compliance, and export controls.

Predix is a platform which means that multitudes of developers can participate in building, sharing and supporting applications; the more participants in the ecosystem, the more value our customers get out of the platform. Predix is creating a portfolio of services for the industrial internet offered both by GE and its development partners. Because it is built on top of Cloud Foundry, any technology vendor can create services for consumption on the Predix platform.



Benefits

- Predix is an **end-to-end platform** that is already built
 - No need to solve typical foundation problems
 - Portfolio of core services for building apps
- Modern software architecture based **on open-source** components
 - Developers have a choice of languages
- Connectivity
 - Secure asset data transfer from edge to cloud



© 2016 General Electric Company - All rights reserved.

10

Predix offers an **end-to-end platform** which means developers can build solutions with a faster time to market without having to worry about the low-level “plumbing” functionality aspects. Developers are relieved from time consuming integration tasks, such as building software server stacks, integrating and configuring products, systems and ‘things’, managing SLAs (service level agreements), and scaling and securing infrastructure.

Within the Predix platform is a **modern software architecture** that is based on open source components that let developers have choice over what development tools (languages) they use.

Predix offers **Connectivity services** to transfer data from edge devices to Predix Cloud for further analysis. It provides a secure and global plug-and-play network over various access networks, including cellular, fixed line, and satellite communication.



Predix Features

Asset-centric

- Services to describe assets; identify components and relationships
- Persist data generated by GE and non-GE assets

Edge to Cloud Coverage

- Platform services connect edge assets to Predix cloud
 - Collect and analyze asset data at the edge
 - Securely transfer asset data to Predix cloud for analysis



End-to-End Security

- Manage user and application authentication
- Control access management to asset data

Analytics Framework

- Software catalog for sharing reusable analytics
- Services for testing and deploying custom analytics



© 2016 General Electric Company - All rights reserved.

11

At the center of industrial applications are a company's assets – wind turbines, locomotives, planes, MRIs – machines operating in the field – the edge – that are transmitting data from sensors. The Predix platform provides data management services that can be used to describe your asset, persist data generated by the asset, and provide data storage. Data services can be used to identify components of an aircraft engine, for example, and describe how all those parts are related.

Connect your Assets using **Edge Services** provided by the Predix platform. These services connect and manage GE and non-GE assets, analyze, then feed asset data into Predix cloud.

Services can collect sensor and asset data, analyze it at the edge, then securely respond to changes based on that data. Connectivity services transfer edge data to Predix Cloud for further analysis. Other services provides a single pane-of-glass view of edge devices providing insights into device connection and network health.

Predix provides **end-to-end security** for your industrial solutions. Security services secure asset data and can handle users and access management. User Account and Authentication services are used to authenticate a user or application requesting to use Predix services. The Access Control Service allows application developers to add granular authorization (who can view and modify data) mechanisms to access web applications and services without having to add complex authorization logic to their code.

The Predix platform provides an **analytics framework** with services to simplify the development of advanced business analyses then deploy them to business operations. Services provides a software catalog for sharing reusable analytics across development teams. The cloud-based framework is designed to facilitate deploying analytics into production on the Predix platform. Developers can implement, test, and deploy new combinations (orchestrations) of analytics without the need for custom coding. As business needs evolve and new analytics are developed, these configurations can be readily updated and redeployed.



Predix Industrial Cloud Platform – Use Case



© 2016 General Electric Company - All rights reserved.

12

Predix Machine collects data from sensors

- **Edge analytics** monitors status of locomotives
- Performance, weather, rail conditions
- Anomaly detected? Corrective action taken
- Decrease in unplanned down times

Locomotive data streamed to **Predix Cloud**

- Analyzed against fleet data
 - Detect trends over time
 - Predict failures, optimize performance
- Predix cloud communicates with edge systems

Predix focuses on building a system-wide view of assets from the edge to the cloud. Data gathered from the edge systems – your assets in the field – is sent to the Predix cloud, which can find insights in the information and send data and recommendations back.

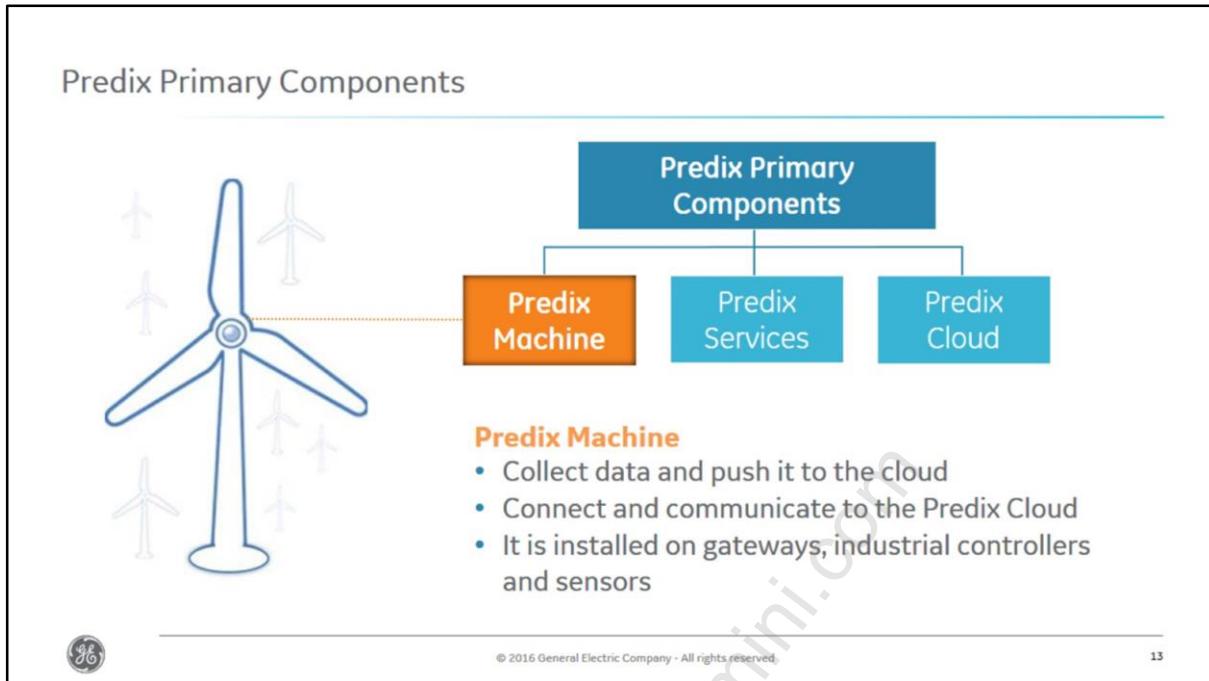
Take the example of a fleet of locomotives.

Beginning at the edge, a single locomotive can have sensors that continuously collect data about the locomotive itself (engine temperature, speed, cargo weight, etc.) as well as its environment – temperature, precipitation, wind speed, and so on. Additionally, video from cameras on the locomotive can show the quality of the track. Predix uses **edge analytics** to collect and analyze this data to monitor the status of the locomotive. If something out of the ordinary is detected, an engineer can take immediate action before a serious problem occurs that could damage the locomotive, pulling it offline for an extended period of time, or cause a serious accident.

Operational data from the locomotive's sensors is used by a Predix application to create a "digital twin" – a computer model – of the physical asset. A constant feed of data about every aspect of the locomotive's operation is streamed to the **Predix cloud**, where that data can be analyzed against operational data collected from other locomotives in the fleet over time.

Each locomotive can benefit from what is happening to other locomotives in the fleet, including rail, environment, weather and engine performance data and factor in that information. For example, if one locomotive reports that a section of track is in poor condition, it can send a message to an application running on the Predix cloud that can tell all the other locomotives to adjust their speed to avoid going too fast.





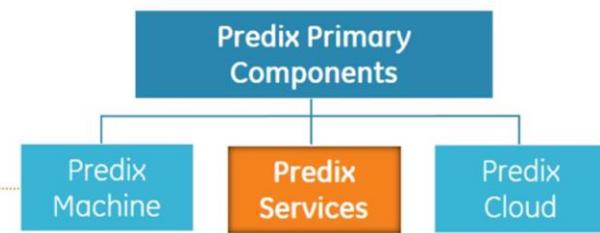
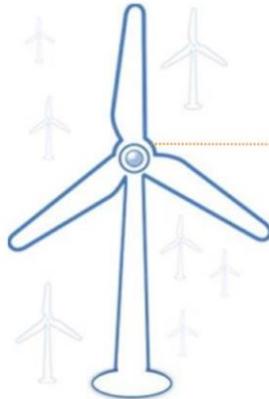
As we take a closer look at the Predix platform, it makes sense to start with the industrial asset, how to connect it to the cloud-based platform and to understand the various services the asset will use. Industrial assets vary in type, location, and service needs. Predix is a comprehensive, modern platform with components that span from the machine to the cloud to enable industrial use cases.

At a high level, the primary components are:

- **Predix Machine:** Predix Machine is the software layer responsible for communicating with the industrial asset and the Predix Cloud, as well as running local applications, like edge analytics. This component can be installed on gateways, industrial controllers and sensors.



Predix Primary Components



Predix Services

- Services that developers can use to build, test, and run industrial internet applications
- Provides a services marketplace
- Developers can publish their own microservices to the cloud
- Optimized for industrial workloads

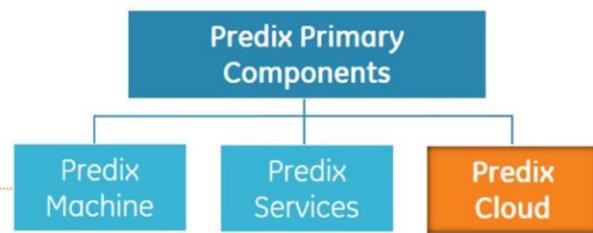
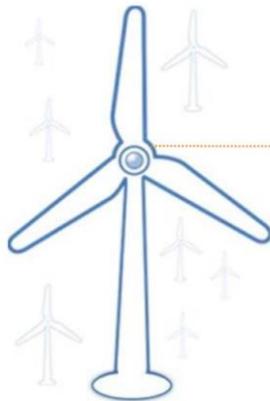
© 2016 General Electric Company - All rights reserved.

14

- **Predix Services:** Predix provides industrial services that developers can use to build, test, and run Industrial Internet applications. It also provides a microservices marketplace where developers can publish their own services as well as consume services from third parties.



Predix Primary Components



Predix Cloud

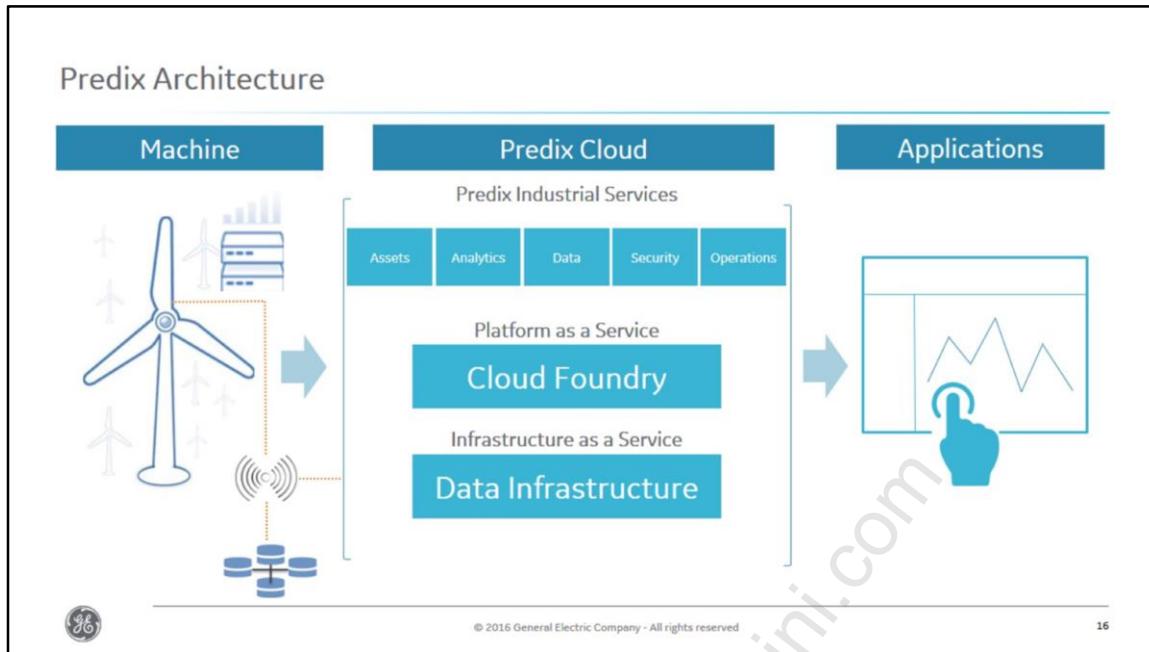
- Global secure cloud infrastructure
- Meet regulatory standards such as healthcare and aviation
- Deploy scalable solutions

© 2016 General Electric Company - All rights reserved.

15

- **Predix Cloud:** The Predix Cloud is a global secure cloud infrastructure that is optimized for industrial workloads and meeting regulatory needs.





The Predix Platform and tools provide a set of software development tools and best practices that enable our customers and partners to optimize their industrial business processes.

Lets take a look at Predix Architecture using the example of predictive maintenance.

Being able to better predict when a problem might occur and proactively shutting down an asset before it causes costly damage or unplanned downtime is just one example of how Predix is being used.

Here we see our industrial asset, the wind turbine. This turbine and the turbine farm sit on the 'edge' of the Predix platform. Using Predix, the data from the wind turbine can be received and analyzed. We can also review and optimize operation to gain maximum value of this asset.

Predix Machine using data collected from the sensors can use edge analytics to monitor the status of an industrial asset. If something out of the ordinary is detected, Predix machine can send an alert to a technician to shut down the asset before something catastrophic happens.

Predix machine can also pass the operational data for this wind turbine and this windfarm to Predix cloud.

With Predix cloud, all data across all windfarms can be stored and analyzed by data scientists. These data scientists can look for trends over time, identify new patters, create new edge analytics and push that information back out to ALL wind turbines.

Predix machine is the software stack that can be installed locally on gateways, industrial controllers, or on sensors. Predix Machine is also responsible for executing edge analytics, and communicating with industrial assets as well as the cloud.

Predix connectivity allows Predix machine to talk to the cloud even when 'normal' internet may not be available.

This can be common in remote locations like oil rigs.

The Predix cloud provides industrial services that developers can use to build test and run industrial internet applications. This is built on a custom built cloud data infrastructure. The data infrastructure is optimized with enhanced security controls and world class data processing and networking capabilities.

Predix Cloud also provides developers an application framework to use industrial microservices as well as a modular design system that allows developers to rapidly deliver innovative applications and a context driven user experience.

From improved analytics, real time asset optimization, or predictive maintenance, the Predix platform is designed to support the continuous improvement of industrial business processes.



Machine/Connectivity

- Predix Machine is a software solution
- Installed locally on industrial controllers, sensors and gateways
- Enables industrial assets to connect to cloud
- Capture, examine, and send the data to cloud for analysis and storage



© 2016 General Electric Company - All rights reserved.

17

- In an industrial setting, you have assets (or physical pieces of hardware) and sensors associated with those assets generating large amount of data.
- Predix Machine makes those assets and sensors “internet ready” by providing software that collects the sensor data and pushes it to the cloud for analytics, storage, and feedback
- Predix Machine can use edge analytics to monitor the status of an industrial asset. If something out of the ordinary is detected, Predix machine can send an alert to a technician to shut down the asset
- With Predix cloud, all data across all windfarms can be stored and analyzed by data scientists. These data scientists can look for trends over time, identify new patterns, create new edge analytics and push that information back out to ALL wind turbines
- Predix connectivity allows Predix machine to talk to the cloud even when ‘normal’ internet may not be available. This can be common in remote locations like oil rigs.



Predix Services

Operational Services	Industrial Services
<p>Manage the lifecycle and commercialization of applications:</p> <ul style="list-style-type: none"> • DevOps Services: develop and deploy applications in the cloud • BizOps Services: enables transparent view of application usage 	<p>Provide core capabilities:</p> <ul style="list-style-type: none"> • Asset Services • Data Services • Analytics Services • Application Security Services • App Services



© 2016 General Electric Company - All rights reserved.

18

Operational Services enables application developers to manage the lifecycle and commercialization of their applications.

- **DevOps Services:** services to develop and deploy industrial internet applications in the cloud
- **BizOps Services:** services that enable transparency into the usage of industrial internet applications so developers can ensure their profitability

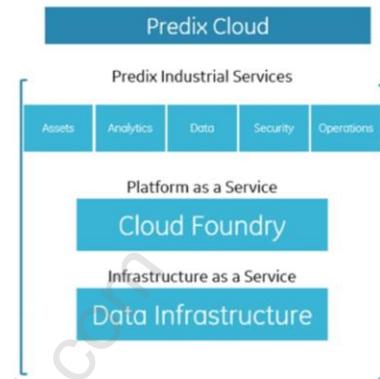
Industrial Services provide the core capabilities required by Industrial Internet applications:

- **Asset Services:** Services to create, import, and organize asset models and their associated business rules
- **Data Services:** Services to ingest, clean, merge, and ultimately store data in the appropriate storage technology so that it can be made available to applications in the manner most suitable to their use case
- **Analytics Services:** Services to create, catalog, and orchestrate analytics that will serve as the basis for applications to create insights about industrial assets
- **Application Security Services:** Services to meet end-to-end security requirements, including those related to authentication and authorization
- **App Services:** Services to enable desktop and mobile views and insights.



Predix Cloud

- Industrial services build, test, and run industrial Internet applications
- Partners around the world build custom cloud infrastructure
- Based on a Software-Defined Infrastructure (SDI)
- Data center evolves with minimal disruption to the applications
- Delivers a shared infrastructure with policy-based provisioning



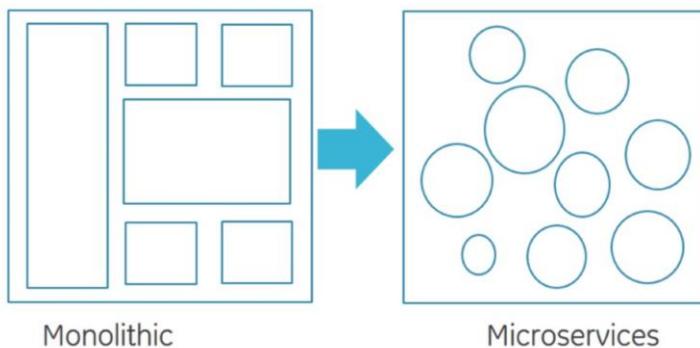
© 2016 General Electric Company - All rights reserved.

19

- The Predix cloud provides industrial **services** that developers can use to build, test, and run industrial Internet applications. It's based on a custom-built cloud data infrastructure
- We carefully specified enhanced security controls and optimized the hardware buildout of our data centers for Predix workloads including world-class data processing and networking capabilities
- GE has carefully specified enhanced security controls and optimized the hardware build-out for Predix workloads, including world-class data processing and networking capabilities
- The result is a GE data center blueprint for meeting industry and regulatory requirements around the globe
- GE has also created a Software-Defined Infrastructure (SDI) that serves as an abstraction layer above the specified hardware, so that the data center can evolve over time with minimal disruption to the applications
- The SDI enables GE to create a shared infrastructure with policy-based provisioning to facilitate dynamic automation and to apply SLA mappings to the underlying infrastructure
- This is especially useful when an application requires an underlying hardware configuration. The provisioning management and pooling of resources can be done at a granular level, allowing for optimal resource allocation and ultimately driving costs down and value up.



Applications



Applications are composed of **small, independent processes** communicating via **language agnostic APIs**

- Focus on doing a single task well
- Scalable
- Reusable
- Easier to Maintain
- Faster delivery of innovation
- Cloud native

CF and microservices together enable Predix to deliver highly scalable industrial solutions



© 2016 General Electric Company - All rights reserved.

20

1. Old days - when we updated – had to bring down entire application to make a small change
2. When app runs as a microservice – just bring down microservice to update and not affect all the other stuff
 - A. Other stuff – like a widget is being updated, rest of app runs, have other data available (other widgets)
3. In the microservice style of software architecture,
 - A. apps are small, independent processes – focus on doing 1 thing well
 - B. Apps (widgets) can be re-used, shared
 - C. Should be able to delivery more quickly
4. Also means you can use language you prefer.

Predix microservices are reusable software modules that can be leveraged as building blocks to rapidly create applications. Because they are developed and delivered as discrete services, these microservices can be loosely coupled into apps without the complexity and dependencies of traditional, monolithic app architectures. Additionally, because microservices can be developed as separate, stand-alone components, developers can use their favorite language and tools.



Predix Cloud

Predix Cloud Basics



© 2016 General Electric Company - All rights reserved.

21

In this module you will log into Cloud Foundry, create a service instance, deploy an app to the cloud and manage your app.



Log into Cloud Foundry: cf login

Identify your API end point:

- Visit predix.io

Enter your login credentials:

- Email ID
- Password

- Predix Basic
<https://api.system.aws-usw02-pr.ice.predix.io>
- Predix Select
<https://api.system.asv-pr.ice.predix.io>
- Predix Japan
<https://api.system.aws-jp01-pr.ice.predix.io>
- Predix UK
<https://api.system.dc-uk01-pr.ice.predix.io>

The API endpoint, or target URL, for your Cloud Foundry instance is the [URL of the Cloud Controller](#) in your Cloud Foundry instance



© 2016 General Electric Company - All rights reserved.

22

When logging into Cloud Foundry, you are prompted for your credentials which include your email address and a password.

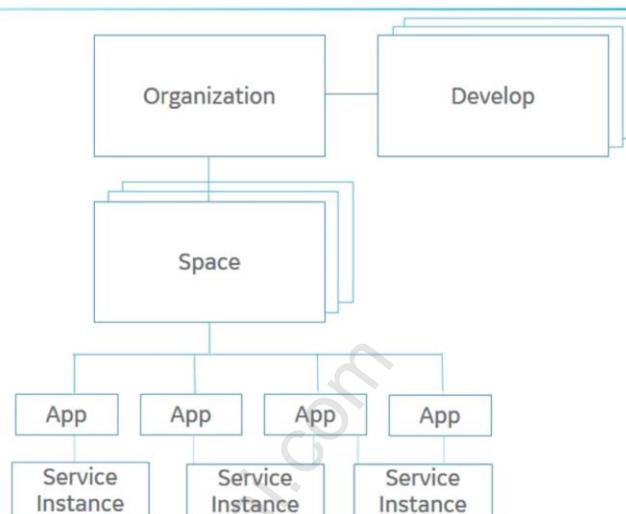
Depending upon your environment, you may be prompted to select an organization and then a space to work in.

For more information: <https://www.predix.io/docs#fSrgvYQd>



Org & Spaces

- An **org** is a development account
- An **org** includes one or more spaces
- A **space** provides access to a shared location
- **Spaces** are used for application development, deployment and maintenance



© 2016 General Electric Company - All rights reserved.

23

When developers log into the Cloud Foundry environment, they have permissions that allow a certain level of access. Your company determines your level of access.

As this graphic indicates, the development environment has organizations which include one or more spaces.

When you log onto Cloud Foundry, you may be able to select an organization and one or more spaces to work in, depending on your permissions.

Organizations are the top-most organizational unit. An example of this is a company name, such as GE.

Spaces sit within an organization, and are the location in which your applications run. This is where you do your development work, deployment, and maintenance of applications.

Applications run within a space and are identified in that space by a Uniform Resource Locator, or URL.

Applications use the services that Predix provides and may be bound to one or more service instance within their space.



Cloud Foundry (cf) Commands

Command	Function
cf login	Login to cloud foundry
cf marketplace	List of available services
cf create-service	Creates a new service instance
cf push	Push an application using manifest.yml
cf bind-service	Binds a deployed app to an existing service instance.
cf scale	Scale app instances up or down
cf target	Find/change orgs & spaces
cf env	List environment variables
cf a	List all apps available within your space
cf s	List all services created within your space



© 2016 General Electric Company - All rights reserved.

24



View Marketplace Services: cf marketplace

Service	plans	description
logstash-X	free	Logstash 1.4 service for application
postgres	shared-nr	Reliable PostgreSQL Service
predix-acs	Tiered	Use this service to provide a more
predix-analytics-catalog	Bronze, Silver*, Gold*	Add analytics to the Predix cloud
predix-analytics-runtime	Bronze, Silver*, Gold*	Use this service to provide a more
predix-analytics-ui	Free	Use this browser-based user interface
Predix-asset	Tiered	Create and store machine asset mode
Predix-blobstore	Tiered	Use this binary large object storage



© 2016 General Electric Company - All rights reserved.

25

- Cloud Foundry offers a marketplace of services, from which users can provision reserved resources on-demand. An example of resources services provide include databases on a shared or dedicated server
- These resources are known as Service Instances and the systems that deliver and operate these resources are known as Services
- After logging into Cloud Foundry you can view what services are available to your targeted organization by running the `cf marketplace` command.



Recognizing Syntax vs. Command Examples

Syntax: Used to denote the elements of a command

Starts with the word syntax

Use of <> brackets between elements

Syntax: cf create-service <service> <plan> <your_name-service_name>

An example of a command

Starts with the directory location

Use of colors (blue, fuchsia, red) to represent the command structure

[predix@localhost ~] \$ cf create-service postgres shared-nr training-postgres

Creating service instance training-postgres in org Predix-Training/space
Training1 as student10 ...

OK

Use of black to represent the code output



© 2016 General Electric Company - All rights reserved.

26

Throughout the course we provide examples of commands as well as their syntax.

Syntax is the example of a command with descriptions of a specific item in a placeholder. These items are contained in <>brackets - seen here in Blue.

A command defines specific items.

In this example we see <service> in the syntax example. In the command example, <service> has been replaced with postgres. When modeling your commands off the syntax, remember not to include the <> brackets



Creating a Service Instance: cf create-service

Syntax: `cf create-service <service> <plan> <your_name-service_name>`

```
[predix@localhost ~] $ cf create-service postgres shared-nr training-postgres
```

```
Creating service instance training-postgres in org Predix-Training/space Training1 as  
student10...
```

```
OK
```

name	service	plan	bound apps	last operation
training-postgres	postgres	shared-nr		create succeeded

Run `cf services` command to display the service instances created in your space



© 2016 General Electric Company - All rights reserved.

27

- The syntax for creating a service instance is: cf create-service, the service name, a plan name, and a service-instance-name
- An instance of the service is created in your Cloud Foundry space. To view a list of service instances in your space, run the `cf services` command.



Predix Cloud

Lab 1: Getting Started with Predix Cloud

</Lab>

- Exercise 1: Logging into Predix Cloud
- Exercise 2: Creating a Service Instance



© 2016 General Electric Company - All rights reserved.

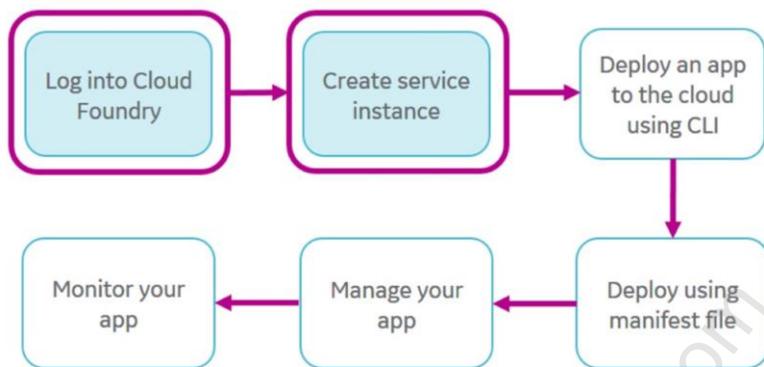
28

Exercise 1: In this exercise you will practice logging in to Predix Cloud

Exercise 2: In this exercise you will use Predix Cloud commands to display all services in the marketplace and create an instance of the postgres service in your Predix Cloud space.



Predix Cloud Lab Process Flow



© 2016 General Electric Company - All rights reserved.

29

This is the first lab and we will be performing initial tasks, such as:

- Login to Cloud Foundry
- Create a service instance



Predix Cloud

Deploy & Monitor a Microservice



© 2016 General Electric Company - All rights reserved.

30



Deploy Microservices to Cloud Foundry

Use `cf push` command to deploy a single application

Syntax: `cf push <your-app-name> -options`

Common option flags:

- b** location of custom buildpack to create a runtime environment
- p** path to application directory where compiled application file is
- m** memory limit
- f** manifest file path

Applications run as microservices inside your space



© 2016 General Electric Company - All rights reserved.

31

Best Practice: Since the options that are specified with the push command can get quite complex, save all of your options in a manifest file to configure your push instead. This section demonstrates how to use a manifest, or configuration file to push your applications.



Deploying Microservices

Demo



© 2016 General Electric Company - All rights reserved.

32

Note: Demo 'Deploying Microservices'.



GE Digital Predix Training

Module: Predix Cloud

32

Deploy Applications to Predix Cloud Using the Manifest

```
[predix@localhost alarmservice]$ cf push
Using manifest file /predix/Documents/MyProject/alarmservice/manifest.yml
Updating app student10-predix-alarmservice in org Predix-Training / space Training5 as student10...
OK

Uploading student10-predix-alarmservice...
Uploading app files from: /predix/Documents/MyProject/alarmservice/target/alarmservice-0.0.1-SNAPSHOT.jar
Uploading 925.5K, 124 files
Done uploading
OK
Binding service training-postgres to app student10-predix-alarmservice in org Predix-Training / space Training5 as student10...
OK

Starting app student10-predix-alarmservice in org Predix-Training / space Training5 as student10...
----> Downloaded app package (24M)
----> Java Buildpack Version: v3.5.1 | http://github.com/pivotal-cf/pcf-java-buildpack.git#d6c19f8
----> Downloading Open Jdk JRE 1.8.0_73 from https://download.run.pivotal.io/openjdk/trusty/x86_64/openjdk-1.8.0_73.tar.gz (1.2s)
----> Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.1s)
----> Downloading Open JDK Like Memory Calculator 2.0.1_RELEASE from https://download.run.pivotal.io/memory-calculator/trusty/x86_64/m
s)
Memory Settings: -XX:MaxMetaspaceSize=104857K -Xss1M -Xms768M -XX:MetaspaceSize=104857K -Xmx768M
----> Downloading Spring Auto Reconfiguration 1.10.0_RELEASE from https://download.run.pivotal.io/auto-reconfiguration/auto-reconfigur
----> Uploading droplet (69M)

Droplet is built and sent to Warden container
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
1 of 1 instances running

App started

OK
```

Uploads the application file to the cloud

Detects and calls the appropriate Buildpack

© 2016 General Electric Company - All rights reserved.

33



Deploying Microservices to Predix Cloud

Showing health and status for app student10-predix-alarmservice in org Predix-Training / space Training1 as student1...

OK

Requested state: started

Instances: 1/1

Usage: 1G x 1 instances

urls: student10-predix-alarmservice.run.aw-usw02-pr.ice.predix.io

last uploaded: Mon July 11 02:12:16 UTC 2016

stack: cflinuxfs2

buildpack: java_buildpack

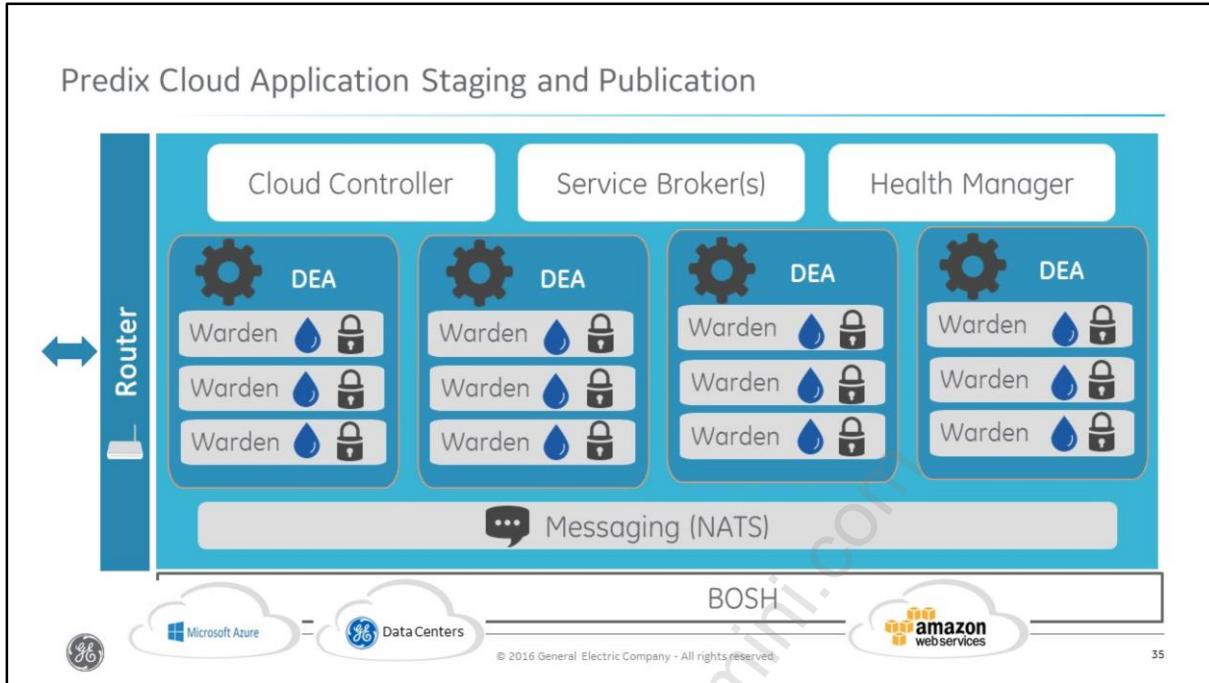
The health manager determines if application has started successfully



© 2016 General Electric Company - All rights reserved.

34





Buildpacks are used in the Cloud Foundry application publication process.

This diagram describes the elements that are used to manage published applications in Predix Cloud:

- At the bottom of the diagram are virtual machines, which allow developers to version, package, and deploy software in a reproducible manner.
- Above that, BOSH, manages the lifecycle of cloud software. It provisions, deploys, and monitors, and manages software over a vast number of virtual machines. Cloud Foundry can run on different IaaS like GE data centers, Microsoft Azure and Amazon Web Services.
- NATS is indicated as the messaging agent and provides open source, performant, and scalable messaging for the Predix Cloud platform.
- Droplet Execution Agents, or DEAs, manage the Wardens, which are containers of applications. The applications are represented here as a droplet, which contains everything an application needs in order to run successfully. Droplets contain the application .war file, its configuration, and any required dependencies, such as runtime and frameworks.
- Predix Cloud controller ensures that everything is working together overall.
- The service broker handles binding, and managing service instances by communicating with the droplets.
- The health manager monitors the whole system checks heartbeat, etc.

Predix Cloud

Lab 2: Deploy and Monitor Applications



- Exercise 1: Deploying an Application



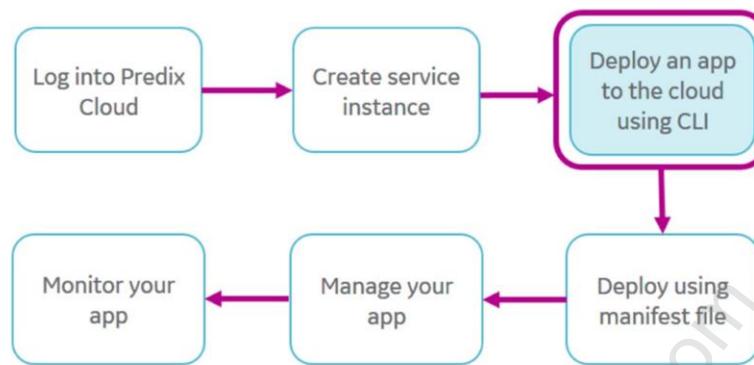
© 2016 General Electric Company - All rights reserved.

36

Exercise 1: In this exercise you will deploy an application to Cloud Foundry from the command line interface.



Predix Cloud Lab Process Flow



© 2016 General Electric Company - All rights reserved.

37



Predix Cloud Manifest File

Describes application deployment options

- Automates subsequent deployments
- Same options as **cf push** command
 - Plus options *only* available in the manifest
- Default name: **manifest.yml**
- Override with **-f** option
 - **cf push -f dev-manifest.yml**

```
manifest.yml
1---
2applications:
3  - name: vs-locomotive-dataingestion-service
4    buildpack: java_buildpack
5    path: target/data-ingestion-0.0.1-SNAPSHOT.jar
6    services:
7      - vs-asset
8      - vs-timeseries
9      - vs-uaa
```



© 2016 General Electric Company - All rights reserved.

38

Manifest files are written in the YML language.

The **cf push** command searches the current directory for a manifest file, unless you specify another path with the **-p** option.

The services line (see screenshot) lists the services to be bound to your application. Using this takes place of the bind services command.

The manifest file allows for more options than the cf push command and a full explanation of these can be found online, but is beyond the scope of this class.



.YML Format

Three dashes indicate start of document

Indent with spaces – not tabs!

- Each indent is two spaces
- “-” defines a group
 - Syntax: **property: value**
 - **#** starts a one-line comment

```
---
Applications:
  - name: nodetestdh01
    memory: 64M
    instances: 2
    path: ...
# comment

  - name: nextapp # # group 2
    memory: 256M
...
...
```



© 2016 General Electric Company - All rights reserved.

39

- Here is a sample manifest file that pushes an application to Predix Cloud. Let’s examine the file’s format. The indentation maintains the data structure hierarchy in the file, so must be maintained consistently throughout the file – parallel elements must have the same number of spaces indented
- The manifest file begins with three dashes
- The application name is preceded by a single dash and one space
- The applications block begins with a heading followed by a colon. The heading indicates the parameter and the data after the colon indicates the value or setting of that parameter
- Subsequent lines in the block are indented **two spaces** to align with name
- You can use the **path** attribute to tell Predix Cloud where to find your application. This is generally not necessary when you run cf push from the directory in which an application is located. The command line option that overrides this attribute is -p.



Manifest vs. Command line

A manifest

- Allows for repeatable deployments
- Allows for multiple deployments at 1 time

Command line

- Single application push only
- Development deployment scenario

```
---
Applications:
  - name: nodetestdh01
    memory: 64M
    instances: 2
    path: ...
# comment

  - name: nextapp # # group 2
    memory: 256M
...
...
```

Command Line overrides manifest parameters



© 2016 General Electric Company - All rights reserved.

40

- There are two methods used to push an application to Cloud Foundry
- The manifest file allows you to save the options for later use, perhaps for other applications. The manifest file also allows you to specify a large number of parameters for your push
- If you use parameters in your cf push command that conflict with the manifest file, the command line parameters override the options outlined in the manifest file
- In the example shown here, the manifest file indicates two instances of your application, with 256 megabytes of memory allocated to each. The push command used in the command line interface indicates 8 instances of the application, with 1,042 megabytes of memory allocated to each
- When both the command line and a manifest file are used, the command line interface overrides the manifest file, so in this case, 8 instances with 1,042 megabytes of memory are pushed to Cloud Foundry.



Binding Services

File Edit View Search Terminal Help

Syntax: **cf bind-service <app_name> <service_instance_name>**

```
[predix@predix-devbox Documents]$ cf bind-service student10-spring-music  
student10-logstash  
Binding service student10-logstash to app student10-spring-music in org  
Predix-Training / space Training5 as student10...
```



© 2016 General Electric Company - All rights reserved.

41



Locating Postgres Environment Variables

```
File Edit View Search Terminal Help
"postgres": [
  "credentials": {
    "ID": 0,
    "binding_id": "b21f2a0e-e4e0-4d16-956c-23b4ee24e9bf",
    "database": "dcf1ea6034360447c9234ce1af1c8df50",
    "dsn": "host=10.72.6.143 port=5432 user=ufbe4b9cc7a534041a48f33273e30b034
password=9852b1e0b2b24b2d9950a97a0fd19287 dbname=dcf1ea6034360447c9234ce1af1c8df50
connect_timeout=5 sslmode=disable",
    "host": "10.72.6.143",
    "instance_id": "896d3dd4-58a7-4f9b-a1d8-968f1cfceb47",
    "jdbc_uri": "jdbc:postgresql://10.72.6.143:5432/dcf1ea6034360447c9234ce1af1c8df50?user=ufbe4b9cc7a534041a48f332
73e30b034\u0026password=9852b1e0b2b24b2d9950a97a0fd19287\u0026ssl=false",
    "password": "9852b1e0b2b24b2d9950a97a0fd19287",
    "port": "5432" ...
```



© 2016 General Electric Company - All rights reserved.

42



Cloud Foundry (cf) Commands for Managing Apps

Command	Function
cf restage	Similar to cf push without uploading the application file
cf stop	Stops the microservice
cf start	Starts the microservice
cf scale	-i Instances -k Disk limit -m Memory limit -f restart
cf delete	-r delete mapped route



© 2016 General Electric Company - All rights reserved.

43

For more information, refer to the Predix 'Ref Card – Training'.



Predix Cloud

Lab 2: Deploy and Monitor Applications

</Lab>

- Exercise 2: Use a Manifest file to Deploy an Application
- Exercise 3: Managing your Application



© 2016 General Electric Company - All rights reserved.

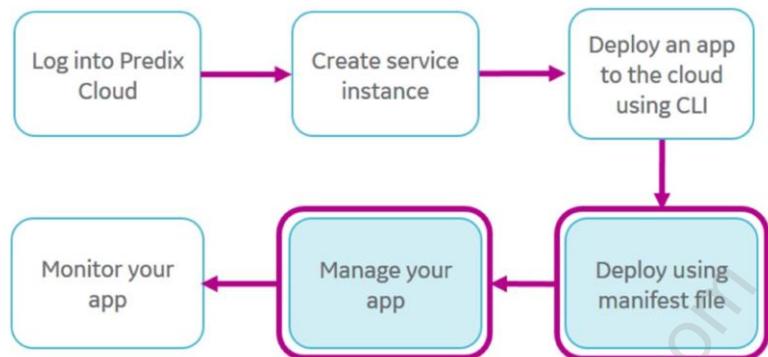
44

Exercise 2: In this exercise you will edit a manifest file and use it to deploy your application instance. The manifest file includes a list of parameters that indicate how the solution should be deployed. Some of the parameters are required, and some are optional. You can also provide these parameters in the command line, but a manifest file is usually used to reduce the complexity of the command, and to save the information for later reuse.

Exercise 3: In this exercise you will environment variables, scale, stop, and delete your application instance.



Predix Cloud Lab Process Flow



© 2016 General Electric Company - All rights reserved.

45



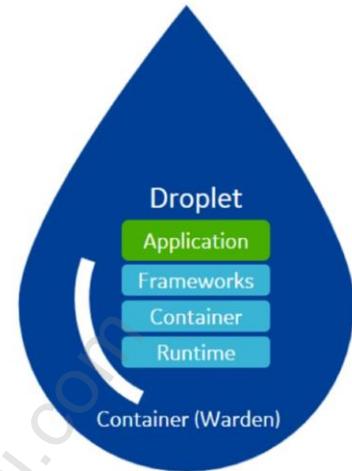
A Buildpack

A combination of scripts that assembles:

- Runtimes
- Containers
- Frameworks
- Your application into a droplet

Typically examines user-provided artifacts to determine:

- What dependencies to download
- How to configure applications to communicate with bound services



© 2016 General Electric Company - All rights reserved.

46

- **Buildpacks** are a convenient way of packaging framework and runtime support for your application
- Buildpacks examine user-provided artifacts to determine what dependencies to download, and how to configure applications to communicate with bound services
- A buildpack builds a droplet that the application runs in
- Applications can be written in different languages and for different platforms, yet they can all be run on Cloud Foundry. This is possible because during a push, Cloud Foundry automatically detects the application's language and provides it on the Droplet Execution Agent (DEA) where the application needs to run.



Buildpack Structure

Ruby script with 3 parts:

1. Detects if the buildpack should be applied
2. Package the droplet by combining the application code with runtimes, frameworks, plugins, etc. necessary for the application (aka Assemble or Pack)
3. Release the app to be deployed to an assigned Droplet Execution Agent (DEA)



© 2016 General Electric Company - All rights reserved.

47

Buildpacks have three main scripts.

1. The detect script determines whether or not to apply the buildpack to an application
2. The compile script builds the droplet that is run by the DEA. It contains all the components necessary to run the application. It is called a compile script, but it is not like a programming language compiler. It simply builds a droplet, not a runtime
3. The release script provides metadata information to Cloud Foundry that indicates how the application should be executed and includes the build location of the app
 - It is also possible to create a custom buildpack for your applications

A few cautions about buildpacks:

- They are not a special build process for your individual application
- Buildpacks build droplets to run on Cloud Foundry
- Buildpacks do not run on a local machine
- They run on Cloud Foundry during the staging process and once the application fully published



Logging and Monitoring your Application

To tail logs

`cf logs <App_Name>`

For most recent logs

`cf logs <App_Name> --recent`

```
2015-08-23T19:12:46.95-0700 [App/0] OUT 02:12:46,954 INFO
SimpleUrlHandlerMapping:315 - Mapped URL path [/assets/**]
onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2015-08-23T19:12:46.95-0700 [App/0] OUT 02:12:46,954 INFO
SimpleUrlHandlerMapping:315 - Mapped URL path [/webjars/**]
onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler] -
```



© 2016 General Electric Company - All rights reserved.

48

The `cf logs` command with your application name, finds the logs for your application. You may run this command from any directory.

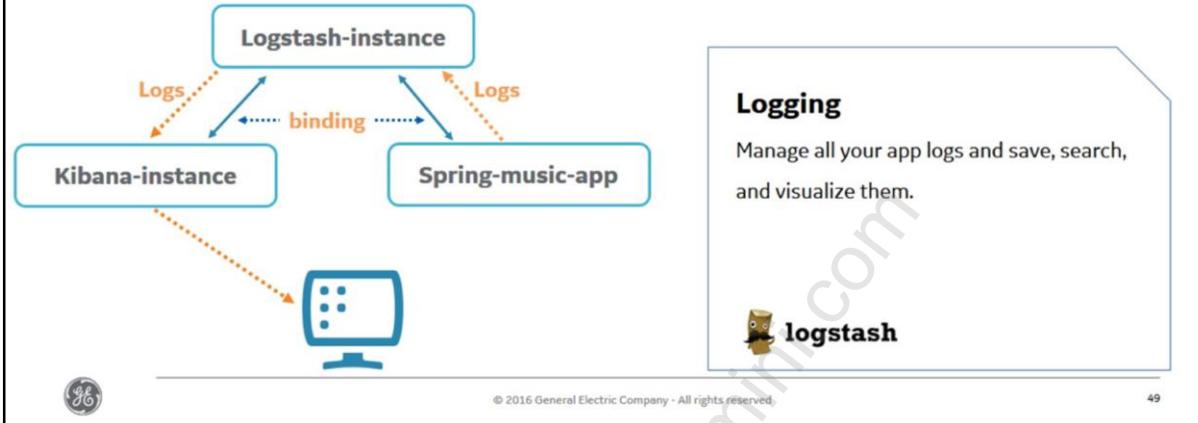
Use the `--recent` option to get the most recent logs available.

You can run the log command in a separate window while you do a push. The log information appears as it the actions occur.



Logging

- Use the **logstash** service view and collate your log files
- **logstash** uses the **kibana-me-logs** application as a web interface into your logs



The diagram shows the following sequence:

1. Create and bind a logging service instance to the spring-music application
2. Create an instance of the Kibana logging service (web application) and bind it to the logging instance
3. Together, these three will provide a web-interface in which you can view spring-music application logs



Predix Cloud

Lab 2: Deploy and Monitor Applications



- Exercise 4: Monitoring your Application



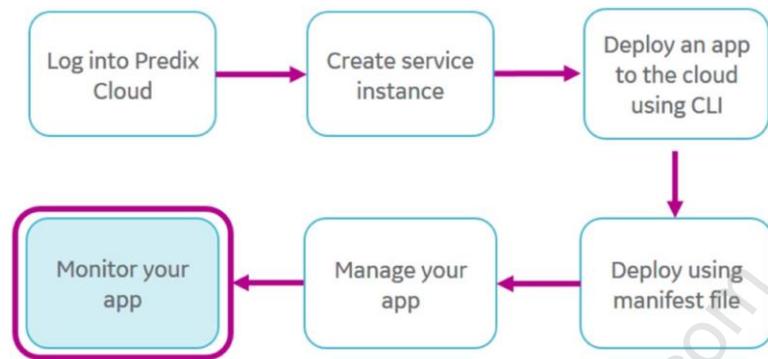
© 2016 General Electric Company - All rights reserved.

50

Exercise 4: In this exercise you will create and bind a logging service instance to the spring-music application. You will also create an instance of the Kibana logging service (web application) and bind that to the logging service instance. Together, these three will provide a web-interface in which you can view spring-music application logs.



Predix Cloud Lab Process Flow



© 2016 General Electric Company - All rights reserved.

51



Predix Cloud

Predix Services Overview



© 2016 General Electric Company - All rights reserved.

52



Data Management Services

Services include:

- Asset Data –create and store machine asset models and instances
- Time Series Store – manage, ingest, store, and analyze
- Blobstore – large byte arrays
- SQL Database – securely store, retrieve data for apps
- Key-value store – redis
- Message Queue (AMQP) - RabbitMQ



© 2016 General Electric Company - All rights reserved.

53

All of these different types of data stores are supported through services in Predix.

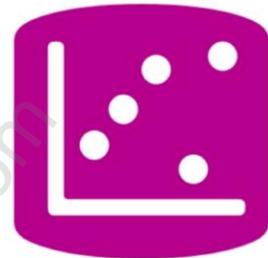
Difference between a relational database like SQL and a time series store:

1. **Relational DB – rows/columns** – good for queries on that type of data – with a unique identifier in one cell (Primary Key)
2. **Time Series store** – stored in column – good for ingesting and searching on that type of data
3. **Time series data** – arrays of numbers indexed by time – aka profiles, curves, traces
 - a. Stock prices – price curve
 - b. Energy consumption – load profile
 - c. Temperature values – temperature trace



Analytics Services

- Analytics Catalog
- Analytics Runtime
- Analytics User Interface



© 2016 General Electric Company - All rights reserved.

54

- Custom analytics – important to do these quickly because code that does analysis can be complex
- Once these are authored by data scientists, they can be re-used
 1. Things like data imputation – (algorithm to replace missing data)
 2. Anomaly detection – something doesn't fit the other data – alert to take a look

Use the analytics catalog service to manage analytic assets authored by data scientists or Predix Analytics you can subscribe to. Developers can configure a classification taxonomy to allow analysts to easily locate analytics for a specific purpose. The Analytics Catalog service supports analytics written in Java, Matlab and Python.

Other analytics services include an analytics runtime, data imputation, GE SmartSignal, and anomaly detection.

Capabilities:

- Develop and deploy **custom analytics** to the cloud
- **Re-usable** analytics maintained in a catalog
- Test and implement orchestrations



Security Services

- User Account Authentication - UAA
- Access Control Service - ACS



© 2016 General Electric Company - All rights reserved.

55

Capabilities:

- Manage user accounts
- OAuth2
- Access Control
- SAML integration with external Identity Provider



Predix Cloud

Lab 3: Building Microservices

</Lab>

- Exercise 1: Deploying a Microservice
- Exercise 2: Adding an Additional API Endpoint to a Microservice



© 2016 General Electric Company - All rights reserved.

56

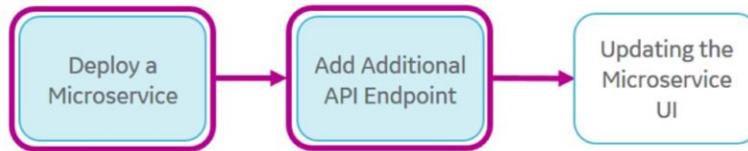
Exercise 1: In this exercise you will build a Java microservice (application) and deploy it to the Predix Cloud.

In this lab we're testing an application locally before deploying it. To test locally, we're starting a local PostgreSQL instance. Once the application is deployed to the Predix Cloud, the local database service is no longer needed.

Exercise 2: In this exercise, you will create another endpoint for the alarmservice microservice. This endpoint will be used to query the hospital table.



Predix Cloud Lab Process Flow



© 2016 General Electric Company - All rights reserved.

57

User Interface

Predix UI – Viewing Data Using Dashboard Seed



© 2016 General Electric Company - All rights reserved.

58



Predix UI

Comprised of

- UX Platform (Px)
- Tools
- Services
- Samples

Benefits

- Structure for modular, layered web apps
- Ease of use
- Responsive to context
- Dashboard seed



© 2016 General Electric Company - All rights reserved.

59

Goal of Predix UI Platform: make the UI development modular, and responsive, context-aware. The Predix UI platform includes User Experience platform, or Px, tools, services, and samples. Developers will get a jump start on creating UI by using the dashboard-style starter pack.



Predix UI Design Elements

Widget – individual containers that displays data (bar chart, etc.)

Card – can contains multiple widgets, elements

Deck –made up of multiple cards

Component –relevant navigation. In our seed app – these are menu navigation items

View – customized content based on context



© 2016 General Electric Company - All rights reserved.

60

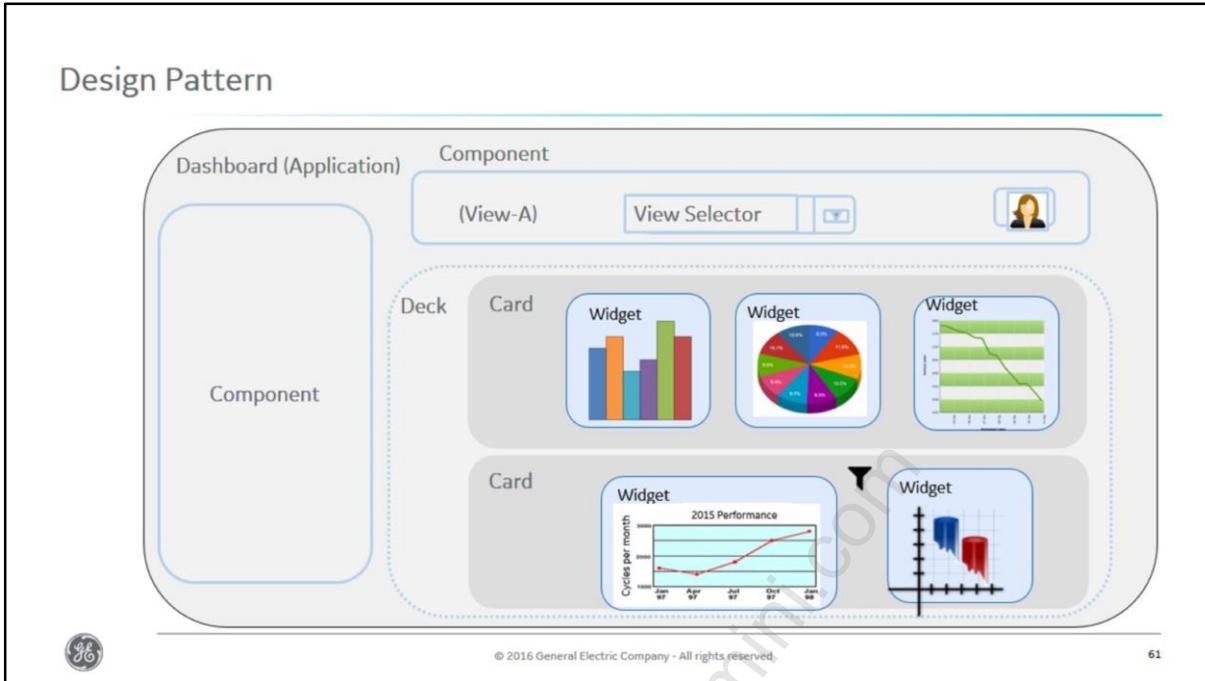
Context defines the scope of the information to be passed through to views, cards and widgets for display.

IMPORTANT: The primary use case for a Context is a particular asset or group of assets to be provided by the Predix Asset service.

View selector shows current view name and opens a list of available views for a given asset context.

This is the important point: Why – because the context browser moves you thru the asset hierarchy. View can be based on logged-on user as well as other things, but the basis of the Predix views is moving through the asset hierarchy.





The Predix UI design philosophy is to make components that are focused, re-usable, self-contained, embrace future standards, and stay compatible.

Context – defines scope of the information to be passed through to views, cards, and widgets for display. Usually think of context based on user, group, role, language, location.
The primary use case for context in Predix is an asset or group of assets.

Main things to understand is that views can be created based on context

Components are containers – include deck, cards, widgets – everything is a container, except for an element – smallest unit on the page

- **Widget** – container for an functional piece of software
- **Card** – contains widgets, elements
- **Deck** –made up of cards
- **Component** – contains other stuff. In our seed app – these are navigation pieces
- **View** – based on context



User Experience Framework: Web Technologies

Polymer – library for creating web elements / components

- <https://www.polymer-project.org/1.0/>
- Install with Bower package manager: <http://bower.io/>
- Polymer helps you build your own custom elements and provides many samples/demos



© 2016 General Electric Company - All rights reserved.

62

The web technologies that the Dashboard starter pack focuses on are Polymer and Sass.

Polymer is a library for creating web elements and components and may be installed using the Bower package manager.

Sass, or syntactically awesome style sheets, is an extension of Cascading Style Sheets, and may be used to provide additional interactivity, simplicity, and functionality to the dashboard.

Let's take a look at each one of these technologies.



Sass

Sass: Syntactically Awesome Style Sheets

Extension of CSS

Compatible with all versions of CSS

<http://sass-lang.com/>



Features:

- Variables – store colors, fonts, CSS values to be re-used
- Allows nesting of CSS selectors similar to HTML nesting
- Partials – CSS snippets available for re-use
- Import – doesn't require additional HTTP request



© 2016 General Electric Company - All rights reserved.

63

Because cascading stylesheets are getting larger and therefore more difficult to maintain, Sass provides additional features that CSS doesn't have.

Developers use Sass to easily create dynamic features and then save their Sass work as a CSS file to use in a web app.

Within Sass, variables store information for reuse so you don't have to keep redefining things like fonts, colors, or other values.

Nesting is not allowed in CSS, but Sass lets you nest CSS selectors so that it follows the same hierarchy as HTML.

Partials are Sass files containing CSS snippets for re-use in other Sass files. This helps you modularize your CSS and eases maintenance.

The import in CSS lets you split CSS into smaller pieces, but splitting this way creates more HTTP requests. Sass combines what you want to import with the file you're importing and serves it as a single CSS file to the browser.

Sass Mixins keep you from having to write CSS declarations repeatedly, and

Sass allows for sharing a set of CSS properties across selectors, which means you don't have to write multiple class names on HTML elements.



Predix UI Component Repositories

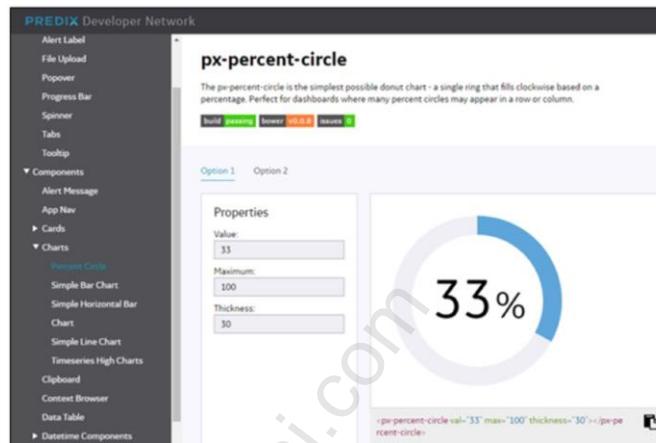
Use Polymer library

Created & maintained by Predix developers

Available on PredixDev github repo:

<http://predixdev.github.io/predix-ui>

- Interactive demos
- API documentation



© 2016 General Electric Company - All rights reserved.

64

Predix UI repositories are available in the PredixDev public github repository at: [github at: http://predixdev.github.io/predix-ui/](http://predixdev.github.io/predix-ui/). Here, users can view live interactive demos and access the API documentation for each component. Links to GitHub allow you to get the working code.

Each Predix UI component is maintained as an individual repository in GitHub (exceptions include px-card and px-chart that are a collection of related components). The Predix UI repositories are created and maintained by Predix developers, and use the Polymer library. You use these repositories, to build web applications that sit on top of the Predix services.



Technologies used to Deploy the Predix Seed App

Technologies used to deploy the Seed App:

- **Bower** – downloads client-side dependency library files
 - bower install
- **npm** – downloads server-side dependency library files
 - npm install
- **GRUNT** – packages files, runs code locally
 - grunt dist
 - grunt serve



© 2016 General Electric Company - All rights reserved.

65

A best practice is to download modify the Predix Seed App to meet your business requirements.

Need to run these commands/utilities:

bower and npm - download dependency library files

- **Bower** is a package manager for the web. Web sites are made of varies elements – frameworks, libraries, assets, and utilities. Bower manages all these things for you. Bower can manage components that contain HTML, CSS, JavaScript, fonts or even image files. Bower doesn't concatenate or minify code - it just installs the right versions of the packages you need and their dependencies.
- **npm** makes it easy for JavaScript developers to share and reuse code (referred to as packages). A package is just a directory with one or more files in it, that also has a file called "package.json" with some meta data about this package. npm is automatically included when Node.js is installed.
- **Grunt** is an automation tool for performing repetitive tasks like minification, compilation, unit testing, linting, etc.

To install server-side and client-side libraries:

- bower install
- npm install
- bower -backend (server-side) libraries
- npm -front-end libraries (browser/client-side)

To package up all files so can be run on a server:

- grunt dist
- grunt serve - starts a local webserver (pops up a browser and goes to localhost:9000)



Predix Cloud

Lab 3: Building Microservices

- Exercise 3: Updating the Microservice UI



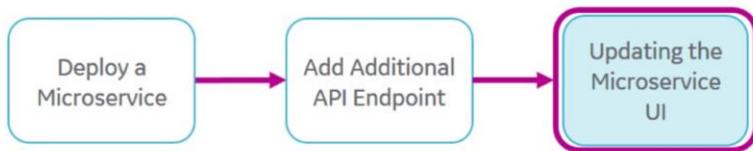
© 2016 General Electric Company - All rights reserved.

66

Exercise 3: In this exercise you will create a UI microservice to display the data fetched by the alarm service. You will use the dashboard seed service pack and modify it to create the UI microservice.



Predix Cloud Lab Process Flow



© 2016 General Electric Company - All rights reserved.

67

Predix Cloud/User Interface: Module Summary

- Developers use Predix services to build, test, and run Industrial Internet applications
- Microservices can be deployed to Predix Cloud using the CLI or a manifest file
- Buildpacks package framework and runtime support for applications
- Predix UI platform includes User Experience platform, or Px, tools, services, and samples



© 2016 General Electric Company - All rights reserved.

68



Security



© 2016 General Electric Company - All rights reserved.

69



Module Overview: Security



© 2016 General Electric Company - All rights reserved.

70



Module Objectives: Security

By the end of this module you will be able to:

- Identify Predix Security services
- Describe the Predix Security architecture
- Explain the UAA authentication options
- Define ACS attributes and policies

Completion of the following lab activities will reinforce the concepts covered in this module:

- Lab 4: Implementing Security in Predix



© 2016 General Electric Company - All rights reserved.

71



Predix Security Services

- **UAA – User Account and Authentication Service**
 - Used to authorize apps to do things on behalf of users
 - Used to authenticate users
 - Capable of integrating with third-party identity providers
- **ACS - Access Control Services**
 - Provides fine-grained authorization
 - Attribute Management
 - Policy Management
 - Policy Evaluation



© 2016 General Electric Company - All rights reserved.

72

- Using existing Security services saves time and resources, so developers do not have to add complex authorization logic to their code
- User Account and Authentication (UAA) provides a service for developers to authenticate their application users
- As a Predix platform user, you can secure access to your application by obtaining a UAA instance from the Cloud Foundry marketplace and configuring it to authenticate trusted users
- All Predix platform services require a UAA instance to ensure secure use of each service

UAA includes the following features:

- Identity management through SCIM APIs, or UAAC
- Login and logout services for UAA authentication
- A complete OAuth 2.0 authorization server
- SAML 2.0 federation capabilities to meet third-party SAML identity provider requirements
- Purpose in life is to create a JWT (JSON (JavaScript Object Notation) Web Token (JWT)) for Authentication and possibly for Authorization.



User Account Management

- [SCIM-based RESTful API](#)
- User native accounts
- 3rd party-managed accounts

- [Manage user permissions](#)
 - Group membership
 - High-Level



© 2016 General Electric Company - All rights reserved.

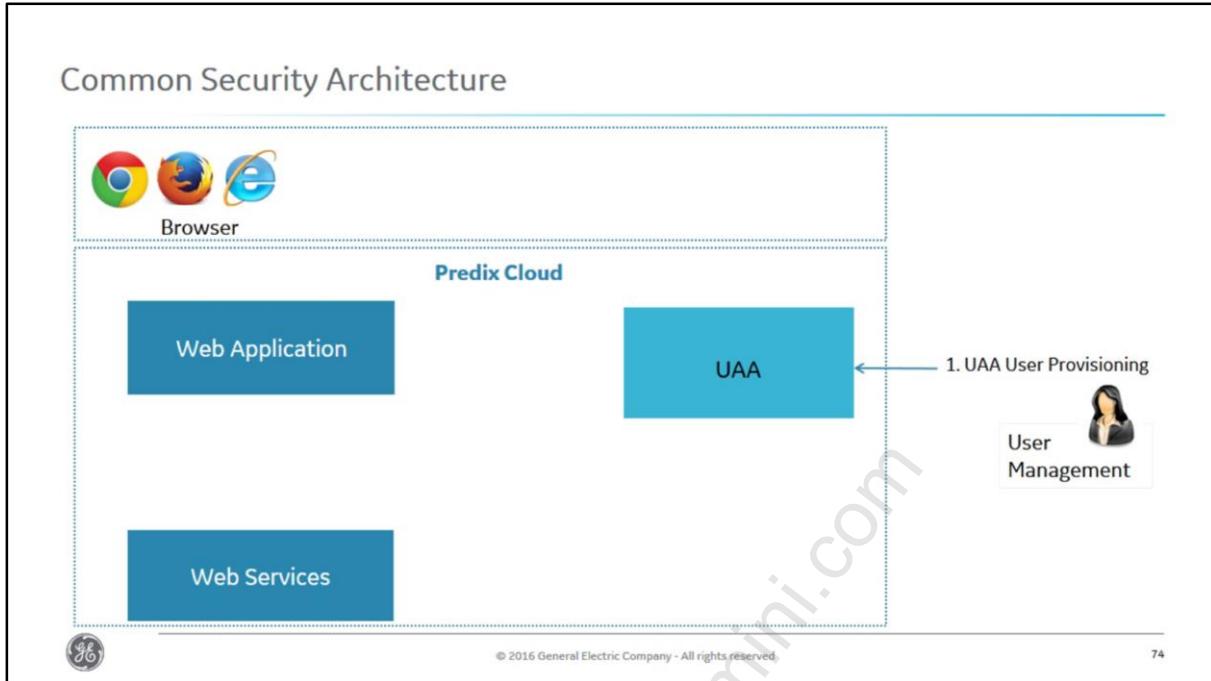
73

Uaa provides a SCIM based REST API to manage user accounts.

User Accounts maybe managed either by adding them to UAA or by trusting a 3rd party provider suck as Ping, OKTA, or computer associates for single sign on.

UAA manages user permissions through group membership and these access permissions are at a higher level than those that can be managed through attributes and policies in the Access Control Service

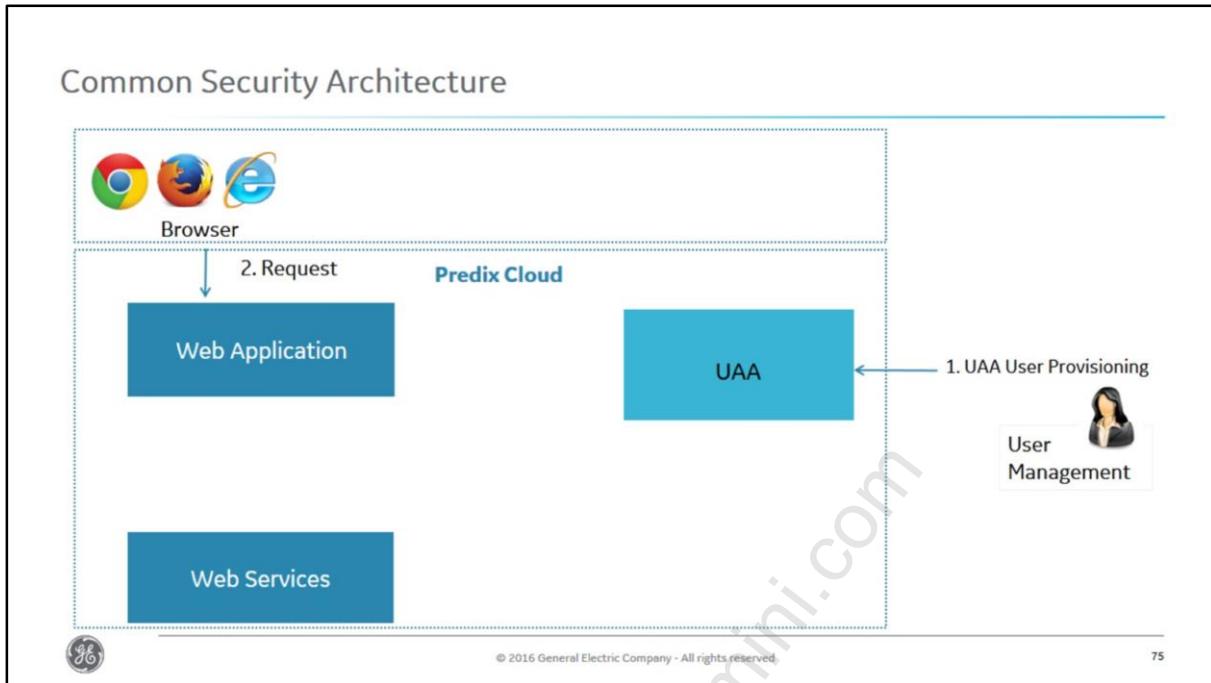




This is a high-level Security Architecture flow, we will also cover a more detailed process.

In this flow:

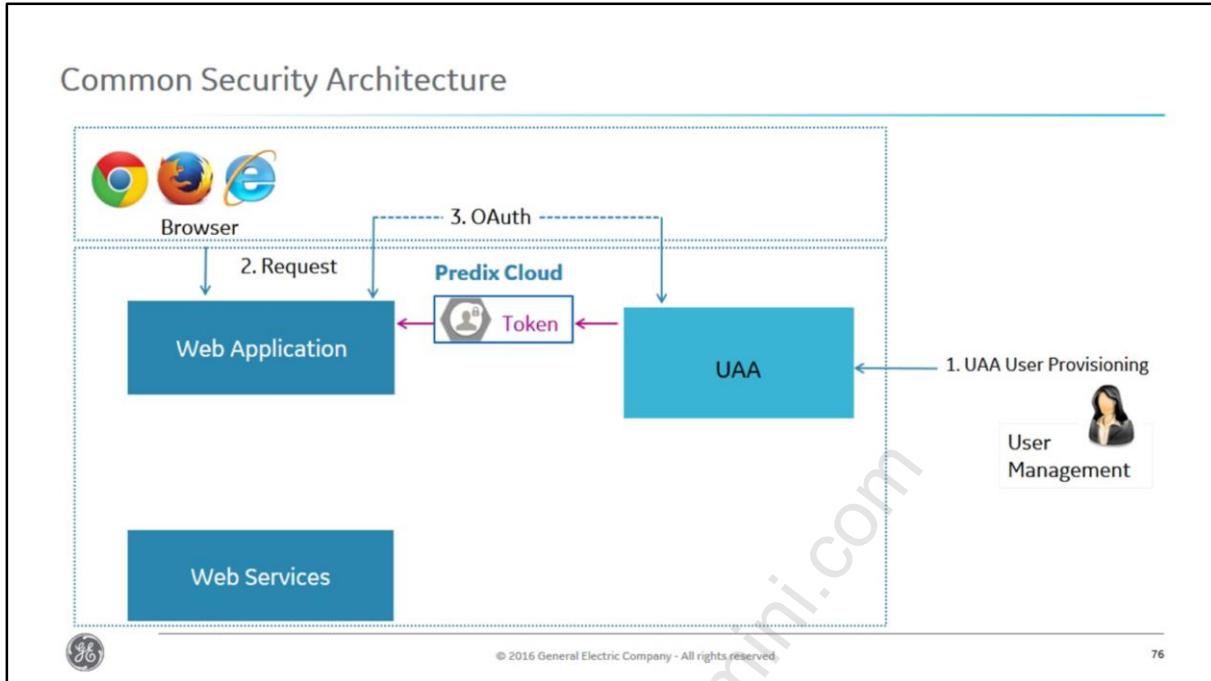
1. An administrator provisions the users through SCIM APIs in UAA
2. An application user requests data using a browser
3. The web application sends the authentication request to UAA. If the user is set up in UAA, the authentication request is approved, the user is authenticated, and the token is issued to the web application. If the data request already contains a valid token, this step is not required
4. The web application passes the data request and JWT token to the web services
5. Once the user is authorized, the data is returned to the web application.



This is a high-level Security Architecture flow, we will also cover a more detailed process.

In this flow:

1. An administrator provisions the users through SCIM APIs in UAA
2. An application user requests data using a browser
3. The web application sends the authentication request to UAA. If the user is set up in UAA, the authentication request is approved, the user is authenticated, and the token is issued to the web application. If the data request already contains a valid token, this step is not required
4. The web application passes the data request and JWT token to the web services
5. Once the user is authorized, the data is returned to the web application.

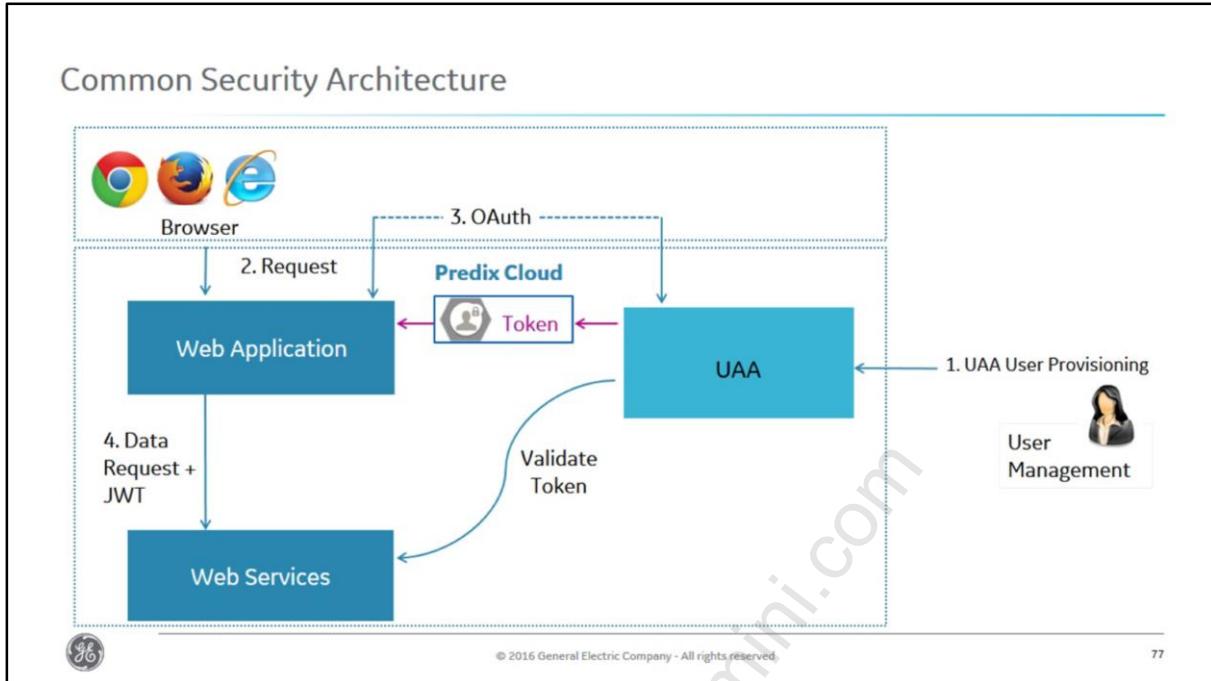


This is a high-level Security Architecture flow, we will also cover a more detailed process.

In this flow:

1. An administrator provisions the users through SCIM APIs in UAA
2. An application user requests data using a browser
3. The web application sends the authentication request to UAA. If the user is set up in UAA, the authentication request is approved, the user is authenticated, and the token is issued to the web application. If the data request already contains a valid token, this step is not required
4. The web application passes the data request and JWT token to the web services
5. Once the user is authorized, the data is returned to the web application.

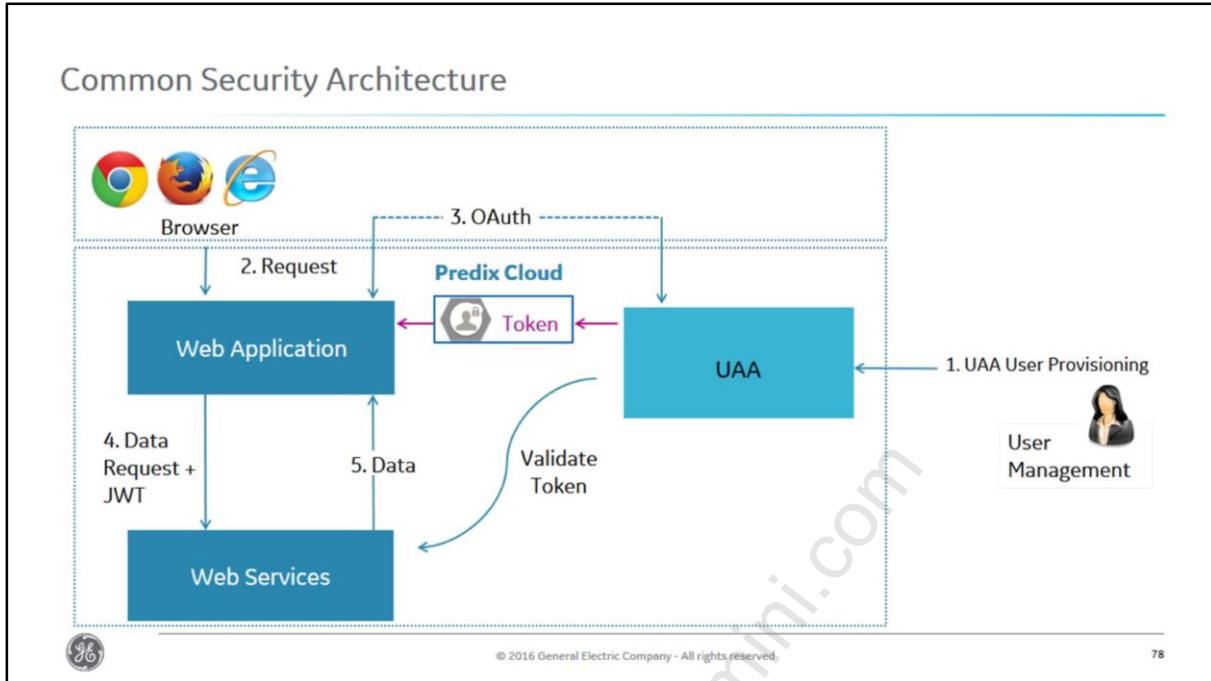




This is a high-level Security Architecture flow, we will also cover a more detailed process.

In this flow:

1. An administrator provisions the users through SCIM APIs in UAA
2. An application user requests data using a browser
3. The web application sends the authentication request to UAA. If the user is set up in UAA, the authentication request is approved, the user is authenticated, and the token is issued to the web application. If the data request already contains a valid token, this step is not required
4. The web application passes the data request and JWT token to the web services
5. Once the user is authorized, the data is returned to the web application.

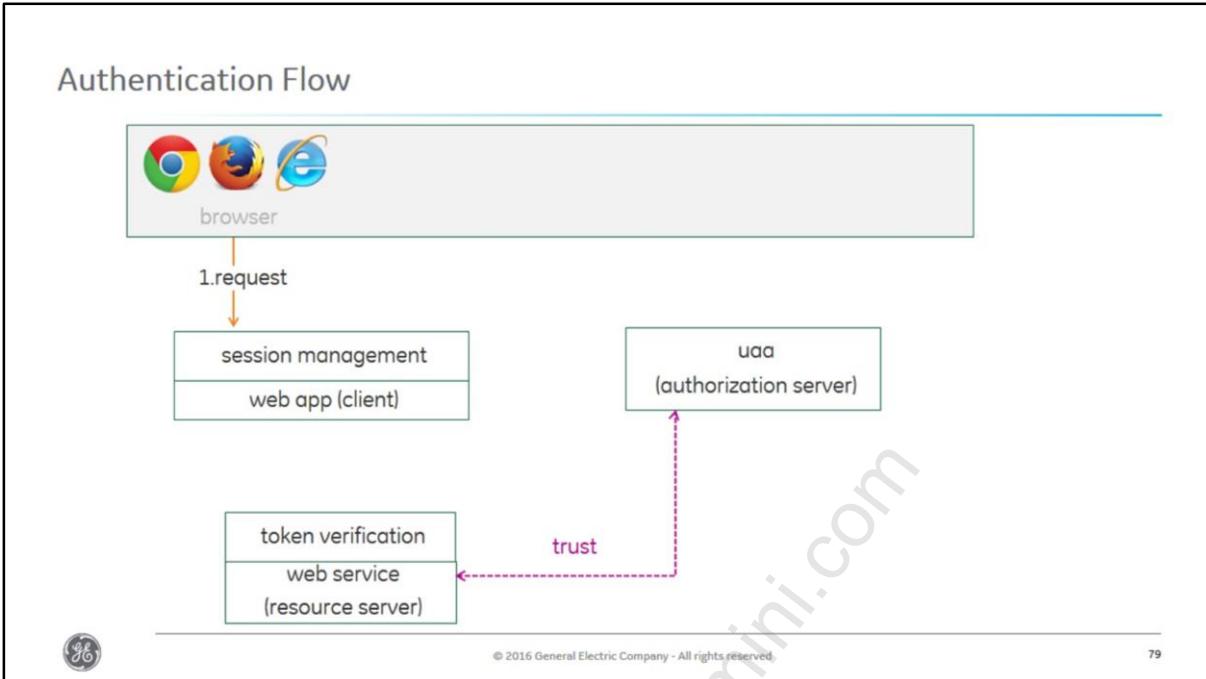


This is a high-level Security Architecture flow, we will also cover a more detailed process.

In this flow:

1. An administrator provisions the users through SCIM APIs in UAA
2. An application user requests data using a browser
3. The web application sends the authentication request to UAA. If the user is set up in UAA, the authentication request is approved, the user is authenticated, and the token is issued to the web application. If the data request already contains a valid token, this step is not required
4. The web application passes the data request and JWT token to the web services
5. Once the user is authorized, the data is returned to the web application.



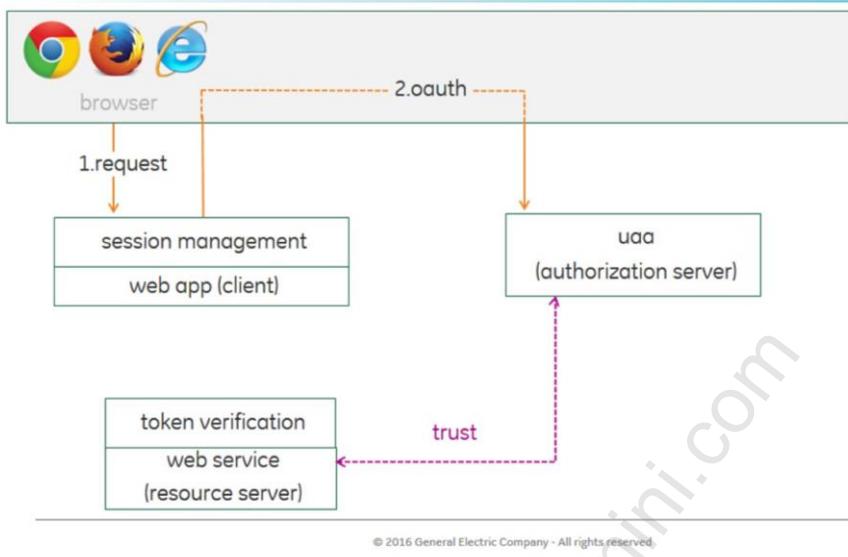


Let's examine the OAuth2 authentication process in more detail:

1. Start with a user making a request to a web application (the “client”) through their browser. The client is responsible for session management of users.
2. The request goes to UAA which is your OAuth2 server
3. The user logs in and is authenticated by the UAA server
4. A code is returned to the app through the user’s browser. The code is only understood by UAA – it is meaningless to the user.
5. & 6. The web app exchanges the code for a token
7. The app can now use the token to request data from a web service
8. If the service verifies the token comes from the right UAA (a trusted relationship must exist between the UAA server and the web service) and contains the necessary permissions, it will return data to the app



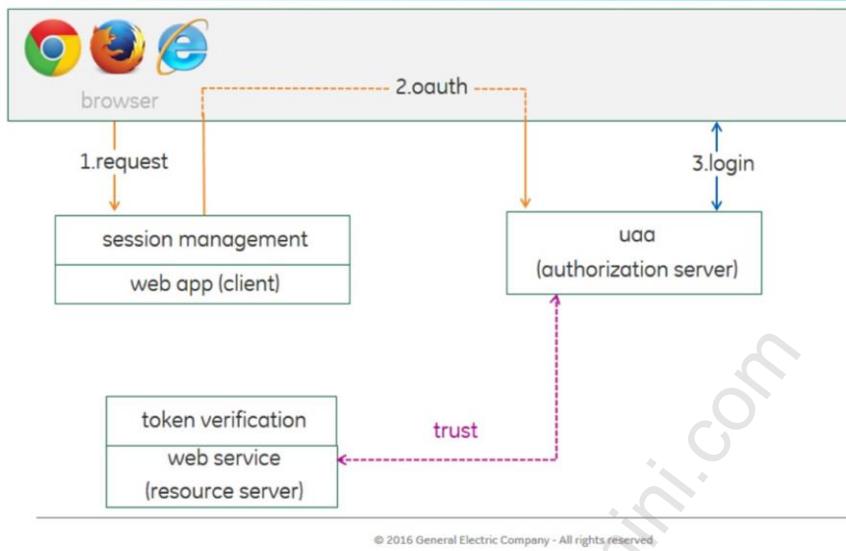
Authentication Flow



Let's examine the OAuth2 authentication process in more detail:

1. Start with a user making a request to a web application (the “client”) through their browser. The client is responsible for session management of users.
2. The request goes to UAA which is your OAuth2 server
3. The user logs in and is authenticated by the UAA server
4. A code is returned to the app through the user’s browser. The code is only understood by UAA – it is meaningless to the user.
5. & 6. The web app exchanges the code for a token
7. The app can now use the token to request data from a web service
8. If the service verifies the token comes from the right UAA (a trusted relationship must exist between the UAA server and the web service) and contains the necessary permissions, it will return data to the app

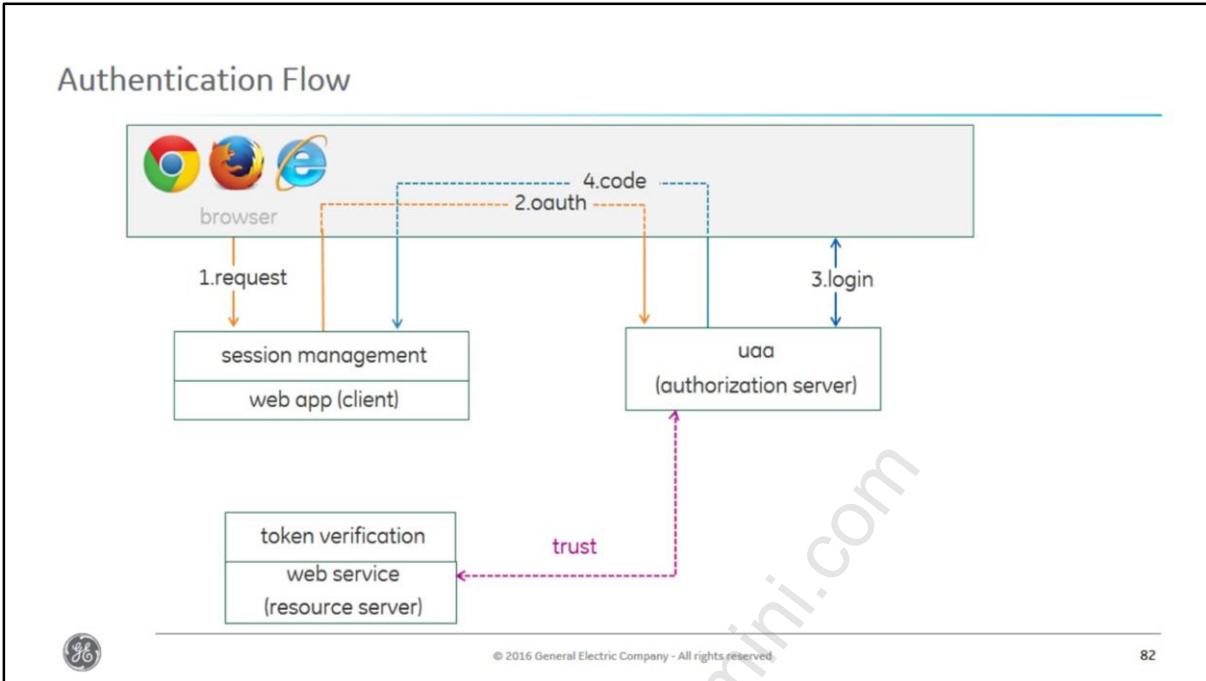
Authentication Flow



Let's examine the OAuth2 authentication process in more detail:

1. Start with a user making a request to a web application (the “client”) through their browser. The client is responsible for session management of users.
2. The request goes to UAA which is your OAuth2 server
3. The user logs in and is authenticated by the UAA server
4. A code is returned to the app through the user’s browser. The code is only understood by UAA – it is meaningless to the user.
5. & 6. The web app exchanges the code for a token
7. The app can now use the token to request data from a web service
8. If the service verifies the token comes from the right UAA (a trusted relationship must exist between the UAA server and the web service) and contains the necessary permissions, it will return data to the app



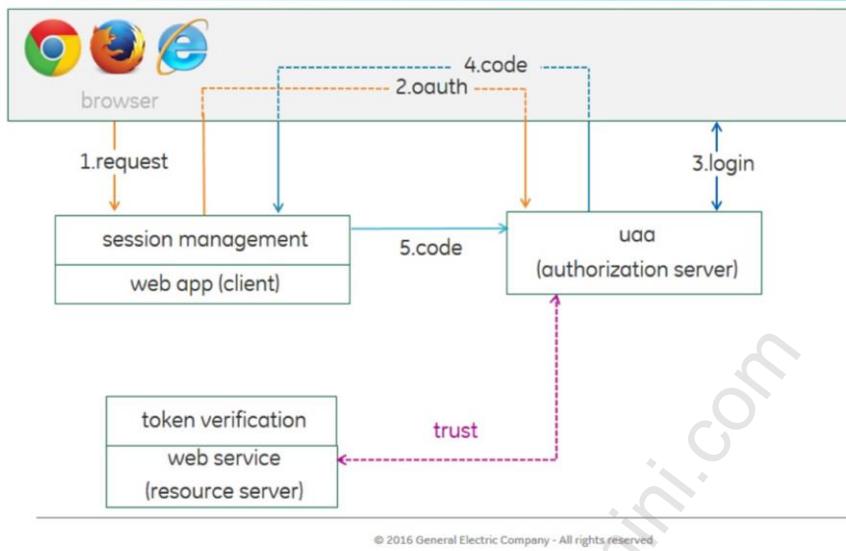


Let's examine the OAuth2 authentication process in more detail:

1. Start with a user making a request to a web application (the “client”) through their browser. The client is responsible for session management of users.
2. The request goes to UAA which is your OAuth2 server
3. The user logs in and is authenticated by the UAA server
4. A code is returned to the app through the user’s browser. The code is only understood by UAA – it is meaningless to the user.
5. & 6. The web app exchanges the code for a token
7. The app can now use the token to request data from a web service
8. If the service verifies the token comes from the right UAA (a trusted relationship must exist between the UAA server and the web service) and contains the necessary permissions, it will return data to the app



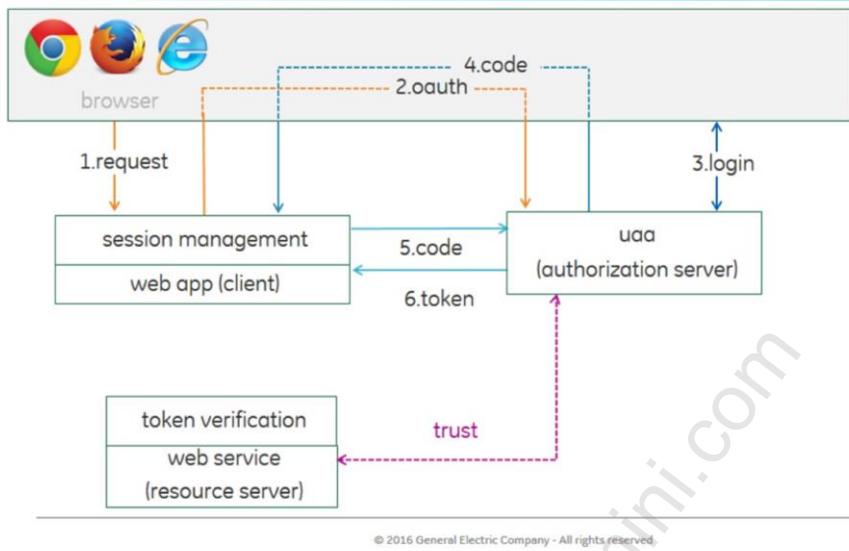
Authentication Flow



Let's examine the OAuth2 authentication process in more detail:

1. Start with a user making a request to a web application (the “client”) through their browser. The client is responsible for session management of users.
2. The request goes to UAA which is your OAuth2 server
3. The user logs in and is authenticated by the UAA server
4. A code is returned to the app through the user’s browser. The code is only understood by UAA – it is meaningless to the user.
5. & 6. The web app exchanges the code for a token
7. The app can now use the token to request data from a web service
8. If the service verifies the token comes from the right UAA (a trusted relationship must exist between the UAA server and the web service) and contains the necessary permissions, it will return data to the app

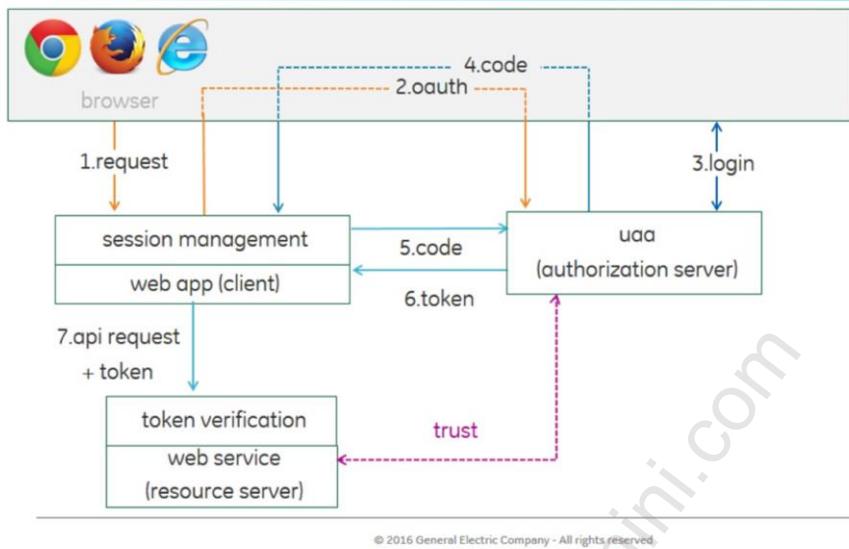
Authentication Flow



Let's examine the OAuth2 authentication process in more detail:

1. Start with a user making a request to a web application (the “client”) through their browser. The client is responsible for session management of users.
2. The request goes to UAA which is your OAuth2 server
3. The user logs in and is authenticated by the UAA server
4. A code is returned to the app through the user’s browser. The code is only understood by UAA – it is meaningless to the user.
5. & 6. The web app exchanges the code for a token
7. The app can now use the token to request data from a web service
8. If the service verifies the token comes from the right UAA (a trusted relationship must exist between the UAA server and the web service) and contains the necessary permissions, it will return data to the app

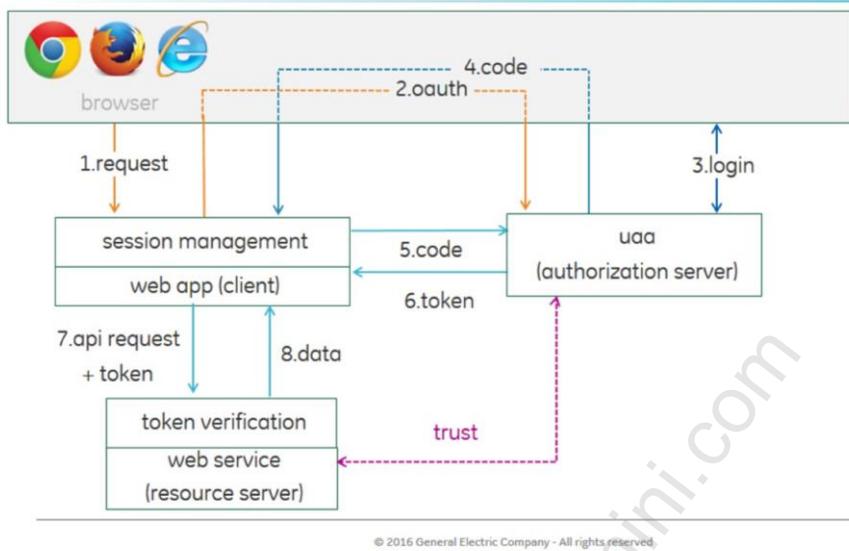
Authentication Flow



Let's examine the OAuth2 authentication process in more detail:

1. Start with a user making a request to a web application (the “client”) through their browser. The client is responsible for session management of users.
2. The request goes to UAA which is your OAuth2 server
3. The user logs in and is authenticated by the UAA server
4. A code is returned to the app through the user’s browser. The code is only understood by UAA – it is meaningless to the user.
5. & 6. The web app exchanges the code for a token
7. The app can now use the token to request data from a web service
8. If the service verifies the token comes from the right UAA (a trusted relationship must exist between the UAA server and the web service) and contains the necessary permissions, it will return data to the app

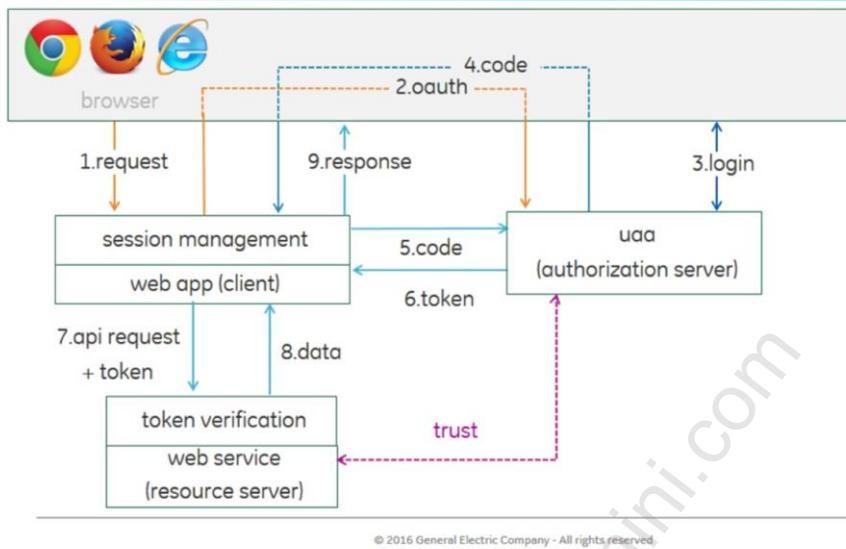
Authentication Flow



Let's examine the OAuth2 authentication process in more detail:

1. Start with a user making a request to a web application (the “client”) through their browser. The client is responsible for session management of users.
2. The request goes to UAA which is your OAuth2 server
3. The user logs in and is authenticated by the UAA server
4. A code is returned to the app through the user’s browser. The code is only understood by UAA – it is meaningless to the user.
5. & 6. The web app exchanges the code for a token
7. The app can now use the token to request data from a web service
8. If the service verifies the token comes from the right UAA (a trusted relationship must exist between the UAA server and the web service) and contains the necessary permissions, it will return data to the app

Authentication Flow



Let's examine the OAuth2 authentication process in more detail:

1. Start with a user making a request to a web application (the “client”) through their browser. The client is responsible for session management of users.
2. The request goes to UAA which is your OAuth2 server
3. The user logs in and is authenticated by the UAA server
4. A code is returned to the app through the user’s browser. The code is only understood by UAA – it is meaningless to the user.
5. & 6. The web app exchanges the code for a token
7. The app can now use the token to request data from a web service
8. If the service verifies the token comes from the right UAA (a trusted relationship must exist between the UAA server and the web service) and contains the necessary permissions, it will return data to the app

Create Your UAA Service Instance

Syntax: `cf create-service predix-uaa-training Free <your_uaa_instance> -c '{"adminClientSecret": "< uaa_admin_secret >"}'`

Annotations:

- `<uaa_admin_secret>"'`: UAA Admin password
- `Free`: Plan
- `<your_uaa_instance>`: UAA instance name
- `-c: in-line JSON object <uaa_admin_secret> in adminClientSecret`



© 2016 General Electric Company - All rights reserved.

88

The next step is to create an predix-uaa service instance. This can also be done through the catalog at predix.io. Use cf marketplace to view the elements. Use the option to provide a file containing service-specific configuration parameters with the path to file:

```
cf create-service predix-uaa <plan> <your_uaa_instance> -c  
'{"adminClientSecret": "<your_secret>"'}
```

The convention used is that Clients have a "secret" and Users have a "password".



Security

Lab 4: Security

</Lab>

- Exercise 1: Create a UAA Service Instance



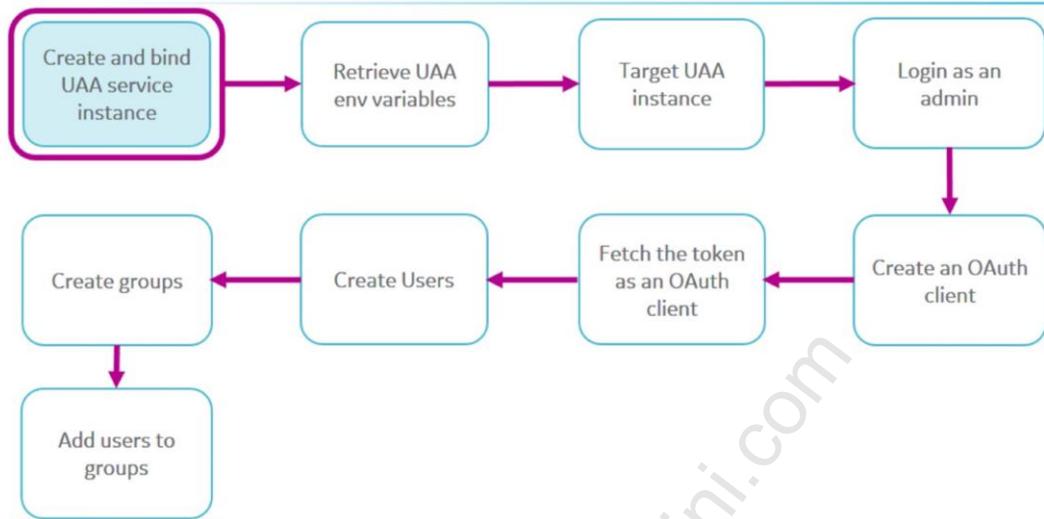
© 2016 General Electric Company - All rights reserved.

89

Exercise 1: In this exercise you will create an UAA service instance and bind your Spring-Music application to that instance. UAA is the Cloud Foundry service that manages users and OAuth2 clients. It is primarily an OAuth2 provider and issues Java web tokens for client applications to use when the applications act on behalf of users. The UAA instance serves as the trusted issuer of tokens and all access to services is provided by authenticating through this trusted issuer.



Security Lab Process Flow



© 2016 General Electric Company - All rights reserved.

90



User, Client & Admin

A **user** is often represented as a **live person**
In UAA, a user belongs to one or more groups



A **client** represents an app that acts on behalf of a user
Best Practice: create a client for each app or service



The **admin client** is the **default client** that has administrative UAA permissions
Automatically created for each UAA instance



© 2016 General Electric Company - All rights reserved.

91

There are some differences in the user, client and admin accounts in UAA

- A **user** is often represented as a live person
- A **client** is an account that acts for itself or on behalf of a user for an application. A client is used to represent your Application.
- Best Practice: create a client for each application or service you create
- The **administrative** client is the default client id that has UAA admin permissions and is automatically created when you create the UAA instance

Once you retrieve the token for admin, you are acting as the UAA administrator with admin privileges. You can then create a client for any Application or service that makes API calls. Each application should have a single Client ID and should not be shared.



OAuth2 Client

- OAuth2 client represents an app
- Specifies client_ID, client secret and authorized grant types

Parameter of a Client	Role
id	Unique client id
scope	Permissions granted to client to act on behalf of a user to access a resource
authorized_grant_types	Determines type of authorization being performed Password, client credentials, authorization code, refresh token
authorities	Permissions granted to client to act on its own behalf to access a resource (no user involvement)
access_token_validity	Token expiration time in milliseconds
redirect_uri	Redirects client to web page after login
autoapprove	Tells UAA which scopes should not require user approval



© 2016 General Electric Company - All rights reserved.

92

The word "client" is in the context of your Application as the Client and the Predix Service is the Server.

You can create additional clients for different applications that authenticate with the same instance of UAA server. When creating a new client, some or all of the following parameters may be used:

Your entire Application has a single **Client Id** and it should not be shared with your users.

Each client has a list of **client authorities** that represent the permissions the client has by itself when acting on its own behalf to access a resource with its own credentials and there is no user involvement.

Scopes are permissions associated with an OAuth Client for a user accessing a resource through an application. The scopes determine if an application is allowed to access on behalf of the user.

An **authorization grant** is a credential representing the resource owner's authorization (to access its protected resources) used by the client to obtain an access token. This specification defines four grant types -- authorization code, implicit, resource owner password credentials, and client credentials

Autoapprove - Tells the UAA that these scopes should not require user approval. When the application (Client ID) opens a new window, the user is not prompted to login again.

After completing its interaction with the resource owner, the authorization server directs the resource owner's user-agent back to the client. Specify a **redirect URI** to redirect client after login. For example, <http://<example-app.com>/welcome>. This URI is used when you start using UAA as service provider for your external Identify provider.



Components of a Token

Contains permissions allowing the Bearer of the token access to services

Parameter	Role
grant_type	Defines type of authorization flow being performed
scope	Defines the permissions that the bearer of the token has
issuer	UAA instance that issues the token
expiration	Tells how long the token is valid



© 2016 General Electric Company - All rights reserved.

93

- When passing information for an authorization request tokens contain much information
- The Authorization grant type defines possible authorization flows a client may use
- The scope that is transmitted in the token defines the range of permissions that this client has been given on behalf of the user
- Authorities define the client permissions it uses on its own behalf

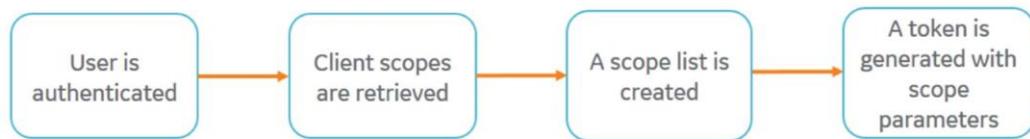
If a token has been defined with AutoApprove, the user will not be prompted each time a new scope or authorization is encountered. If the scope or authorization and auto approve are defined in the token the token will automatically move through scope approvals and the user will not be prompted each time he or she moves to a new page in the application.

Parameters define roles that the user transmitting the token has been assigned.



A Token is Requested: Step-by-Step

When a client requests a token on behalf of a user (authorization_code grant type):



© 2016 General Electric Company - All rights reserved.

94

<https://github.com/cloudfoundry/uaa/blob/master/docs/UAA-Tokens.md>



UAA CLI (UAAC) Commands

Command	Function
uaac help	Display summary or details of command or topic
uaac target	Display current or set new target
uaac token client get <client name>	Gets a token by posting client credentials
uaac users	List user accounts
uaac groups	List groups
uaac clients	List client registrations
uaac context	Display or set current context (who is logged in)



© 2016 General Electric Company - All rights reserved.

95



Retrieve UAA Environment Variables

```

File Edit View Search Terminal Help
"predix-uaa": [
  {
    "credentials": {
      "issuerId": "https://005d03f9-b0a1-4030-993c-7b2db48f9a9e.predix-uaa.run.aws-
usw02-pr.ice.predix.io/oauth/token",
      "subdomain": "005d03f9-b0a1-4030-993c-7b2db48f9a9e",
      "uri": "https://005d03f9-b0a1-4030-993c-7b2db48f9a9e.predix-uaa.run.aws-usw02-
pr.ice.predix.io",
      "zone": {
        "http-header-name": "X-Identity-Zone-Id",
        "http-header-value": "005d03f9-b0a1-4030-993c-7b2db48f9a9e"
      }
    }
  }
]
  
```

Trusted Issuer Id

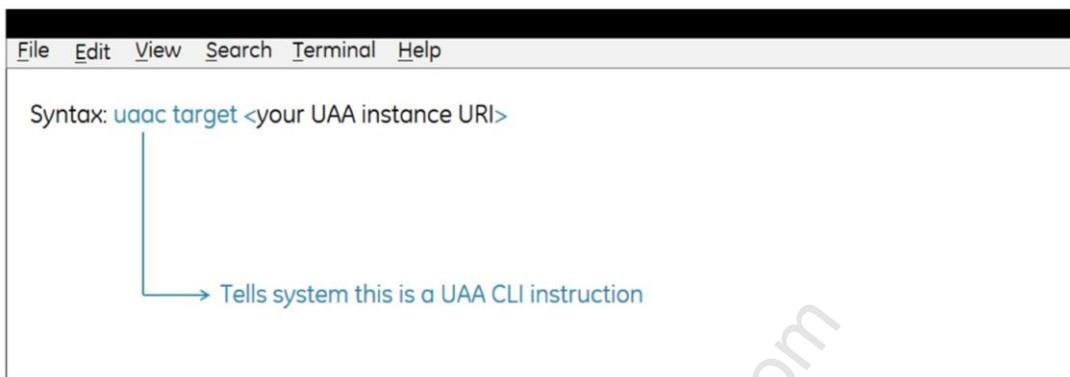
Use the UAA uri to target your UAA instance, so token can be fetched

© 2016 General Electric Company - All rights reserved.

- You need a token to authenticate on behalf of your users, but before you can get this token you have to configure your UAA instance
- You need to identify the UAA instance you want to work with, so you need the UAA instance URL. You can retrieve the UAA instance URL from the VCAP_SERVICES environment variable
- Copy the `uri` value under the `credentials` section, using `cf env <application name>`
- When you bind UAA instance to the Predix App, the connection details are populated in VCAP_SVCS environment variables. These environment variables are used by REST client to call APIs



Target Your UAA Instance



© 2016 General Electric Company - All rights reserved.

97

Specify your UAA instance as the intended target. Paste the **uri** value in the **uaac target <uri>** command.

"uri": "https://<UAA Zone ID>.predix-uaa-training.run.aws-usw02-pr.ice.predix.io",



Login to UAA as Administrator

```
File Edit View Search Terminal Help
```

Syntax: `uaac token client get admin -s <uaa_admin_secret>`

 └── `<uaa_admin_secret>`
 UAA Admin password

 └── `-s <my_secret>`
 Admin client automatically
 created with UAA instance creation

 └── `Fetch the token for the admin client`



© 2016 General Electric Company - All rights reserved.

98

In this step, you will login as the administrator. Admin is automatically created when you create an instance, and can be used to create clients and users. You login as the *admin* client by using a fetch token command.

Tokens hold the authentication and authorization data.

`uaac token client get admin -s <my_secret>`

A successful fetch notice indicates you have retrieved the token.

You are now acting as the UAA administrator with admin privileges. Admin privileges allow you to perform many tasks.

For instance; the value `scim.write` represents sufficient permission to create accounts.



Create an OAuth Client

```
File Edit View Search Terminal Help
OAuth Client      client password
Syntax: uaac client add <client name> -s <client secret>
--authorized_grant_types "<list grant types>"
--authorities "<list client permissions>"
```



© 2016 General Electric Company - All rights reserved.

99



Fetch the Token as an OAuth Client

You create an OAuth Client to represent your app

Syntax: `uaac token client get <client_name> -s <client_secret>`

- `<client_name>` → OAuth client
- `-s <client_secret>` → OAuth client password
- `<client_name>` → Fetch the token for the OAuth Client



© 2016 General Electric Company - All rights reserved.

100

In this step, you will login as the application client.

Best Practice: Do not include the `-s <your_secret>` section, then you will be prompted for it. This way you do not display your password in clear text.



Security

Lab 4: Security

</Lab>

- Exercise 2: Fetch a UAA token



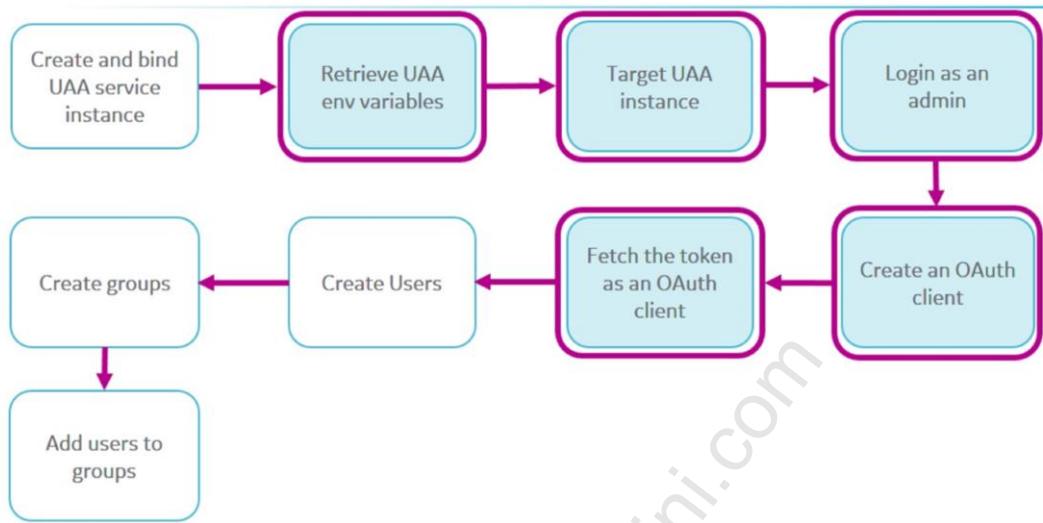
© 2016 General Electric Company - All rights reserved.

101

Exercise 2: In this exercise you will use the UAA Command Line Interface (UAAC) to fetch a token from the UAA service instance created in the previous lab. The UAAC has been installed on your DevBox.



Security Lab Process Flow



© 2016 General Electric Company - All rights reserved.

102



Grant Types in OAuth2

Defines the authorization flow

Grant Types	Description
client_credentials	Service-to-service authentication
refresh_token	Extends authorization by obtaining a new token when the original access token expires
authorization_code	Browser-based end user authentication
password	Used to exchange a username and password for an access token directly



© 2016 General Electric Company - All rights reserved.

103

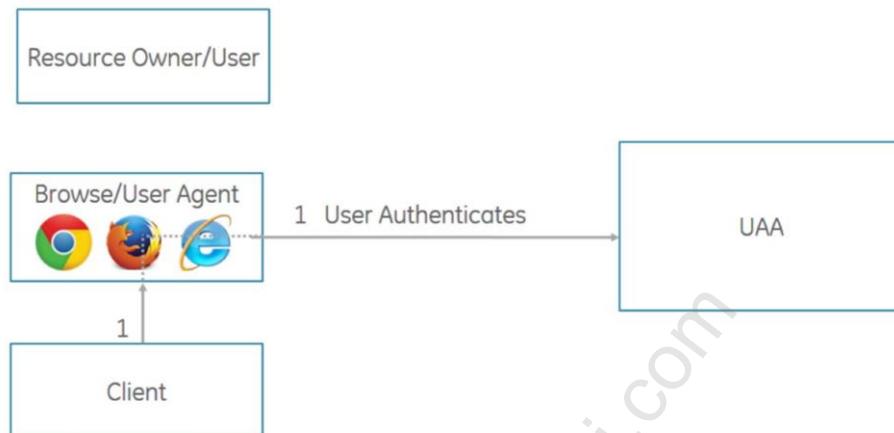
The Authorization code grant type maintains user password confidentiality by sending an authorization code to request a token, rather than a user name or password.

The Client credential grant type does not involve the user; instead the access token is passed only for service calls to the REST web service.

The Extension or Refresh Token grant type fetches a new token once your initial token expires. This grant type can be used when running an analytic that is scheduled to take over 24 hours.



Authorization Code Grant Type



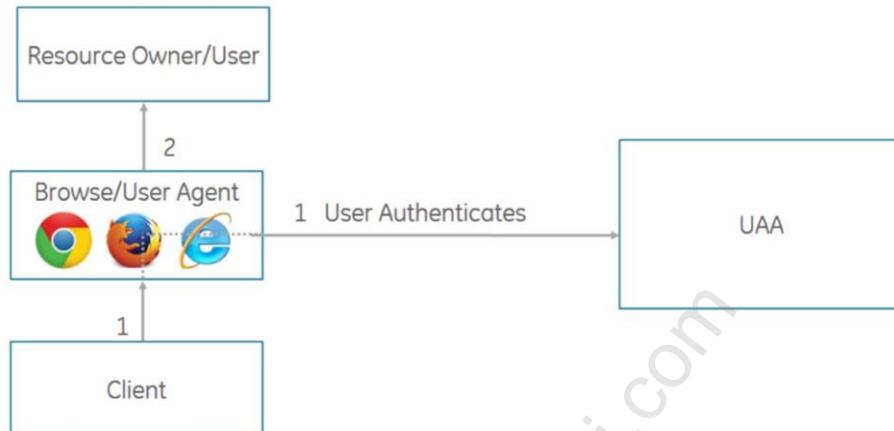
© 2016 General Electric Company - All rights reserved.

104

1. The client requests access to a resource. The request includes client_ID, requested scope, local state, and a redirection URI
2. UAA authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request
3. If the resource owner grants access, UAA sends the authorization code back to the client using the redirection URI provided in previous step
4. The client requests an access token from the UAA token endpoint by including the authorization code received in the previous step
5. UAA authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client. If the client is authenticated, UAA sends an access token back.



Authorization Code Grant Type

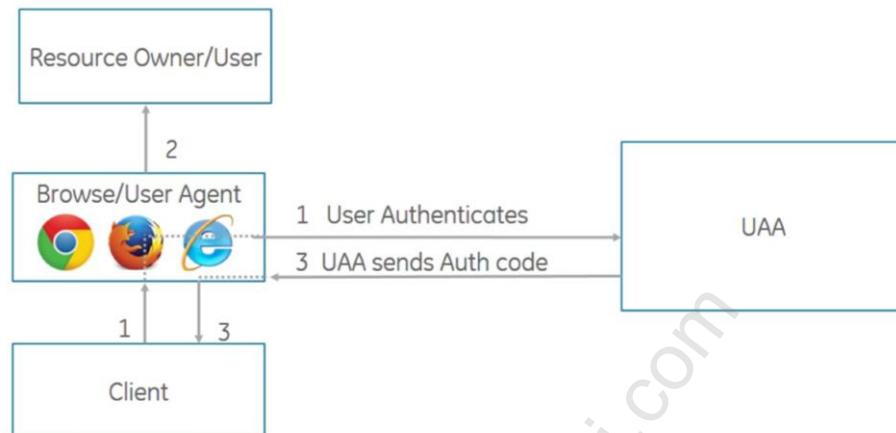


© 2016 General Electric Company - All rights reserved.

105

1. The client requests access to a resource. The request includes client_ID, requested scope, local state, and a redirection URI
2. UAA authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request
3. If the resource owner grants access, UAA sends the authorization code back to the client using the redirection URI provided in previous step
4. The client requests an access token from the UAA token endpoint by including the authorization code received in the previous step
5. UAA authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client. If the client is authenticated, UAA sends an access token back.

Authorization Code Grant Type



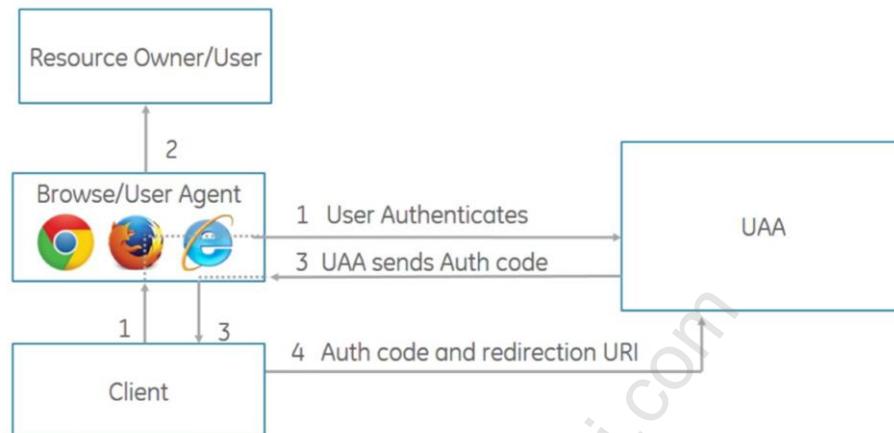
© 2016 General Electric Company - All rights reserved.

106

1. The client requests access to a resource. The request includes client_ID, requested scope, local state, and a redirection URI
2. UAA authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request
3. If the resource owner grants access, UAA sends the authorization code back to the client using the redirection URI provided in previous step
4. The client requests an access token from the UAA token endpoint by including the authorization code received in the previous step
5. UAA authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client. If the client is authenticated, UAA sends an access token back.



Authorization Code Grant Type



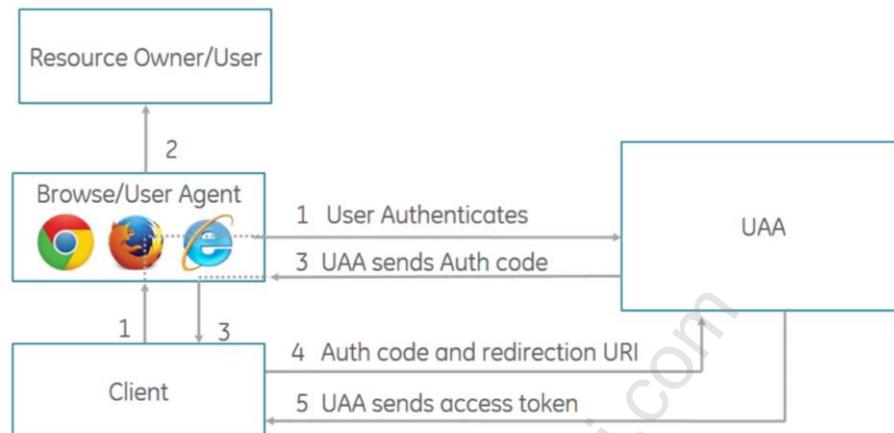
© 2016 General Electric Company - All rights reserved.

107

1. The client requests access to a resource. The request includes client_ID, requested scope, local state, and a redirection URI
2. UAA authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request
3. If the resource owner grants access, UAA sends the authorization code back to the client using the redirection URI provided in previous step
4. The client requests an access token from the UAA token endpoint by including the authorization code received in the previous step
5. UAA authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client. If the client is authenticated, UAA sends an access token back.

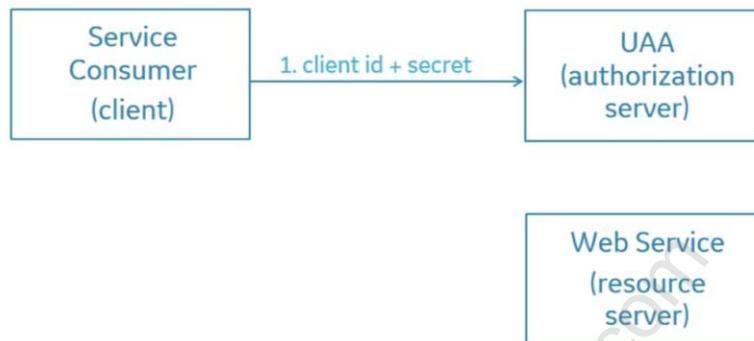


Authorization Code Grant Type



1. The client requests access to a resource. The request includes client_ID, requested scope, local state, and a redirection URI
2. UAA authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request
3. If the resource owner grants access, UAA sends the authorization code back to the client using the redirection URI provided in previous step
4. The client requests an access token from the UAA token endpoint by including the authorization code received in the previous step
5. UAA authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client. If the client is authenticated, UAA sends an access token back.

Client Credential Grant Type



© 2016 General Electric Company - All rights reserved.

109

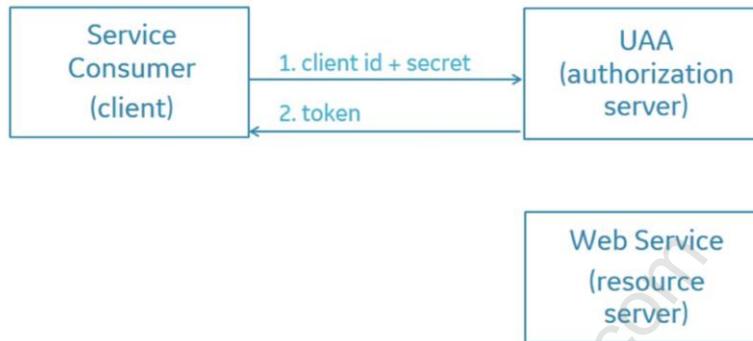
Now we will describe the authorization-flow for the more common use-cases of these protocols.

The client credentials grant type is used to provide an access token to a client based on the credentials of the client, not the resource owner. In this grant type, the client credentials are swapped for an access token (step 1).

For cases such as; a service consuming another service.



Client Credential Grant Type



© 2016 General Electric Company - All rights reserved.

110

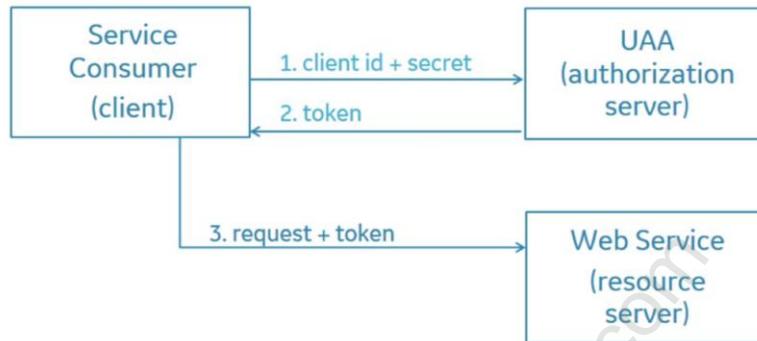
Now we will describe the data-flow for the more common use-cases of these protocols.

The client credentials grant type is used to provide an access token to a client based on the credentials of the client, not the resource owner. In this grant type, the client credentials are swapped for an access token (step 1).

For cases such as; a service consuming another service.



Client Credential Grant Type



© 2016 General Electric Company - All rights reserved.

111

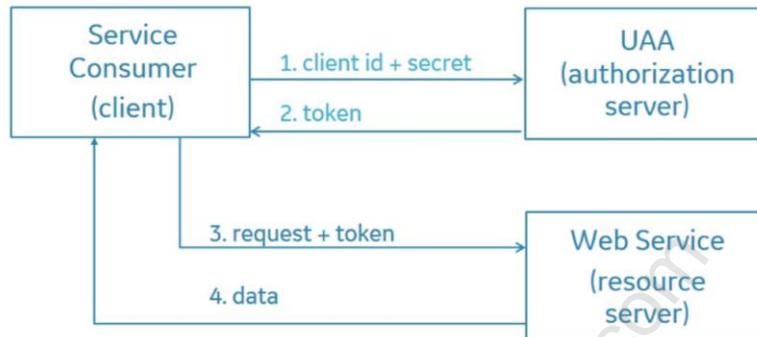
Now we will describe the data-flow for the more common use-cases of these protocols.

The client credentials grant type is used to provide an access token to a client based on the credentials of the client, not the resource owner. In this grant type, the client credentials are swapped for an access token (step 1).

For cases such as; a service consuming another service.



Client Credential Grant Type



© 2016 General Electric Company - All rights reserved.

112

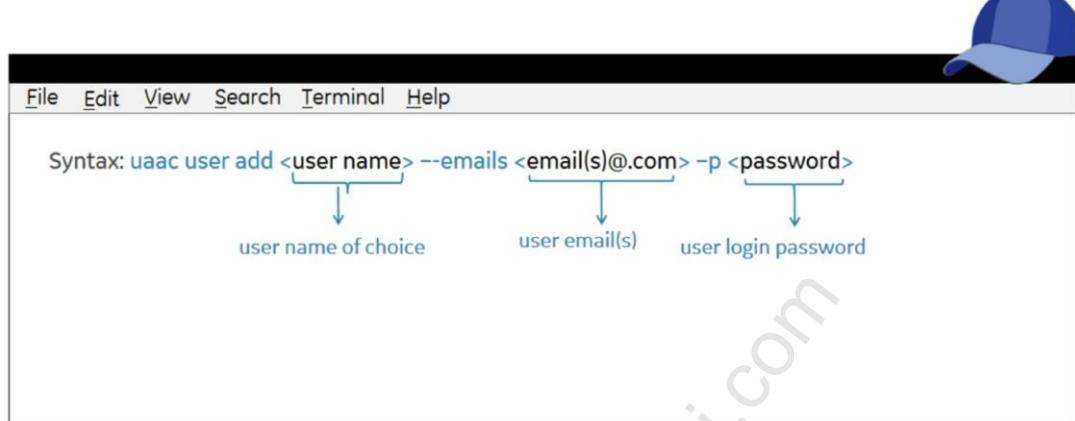
Now we will describe the data-flow for the more common use-cases of these protocols.

The client credentials grant type is used to provide an access token to a client based on the credentials of the client, not the resource owner. In this grant type, the client credentials are swapped for an access token (step 1).

For cases such as; a service consuming another service.



Create Users



© 2016 General Electric Company - All rights reserved.

113



Authorities vs. Groups vs. Scopes

Authorities

- What permissions a client has when it acts on its own behalf

Groups

- What permissions a user **potentially** has
- Effective permissions are still limited by the client scopes



Regardless of whether the client acts on its own behalf or on behalf of a user, you still need to know what the client is allowed to do in each situation i.e. what are its privileges. In UAA, the privilege model for clients and users is governed by authorities, groups, and scopes....

A group is a privilege assigned to a user which is limited by client scopes.

Privileges assigned to a user allow them to **potentially** read and write identity managed information.

Users **potentially** have scim.read and scim.write privileges (not always). Privileges depend ultimately on the plan the users are accessing these resources under and who is delegating access to it (client scopes).

The actual privileges the user has are displayed in the access token issued.

Authorities vs. Groups vs. Scopes

- Client scope
 - Permissions a client has on behalf of a user
 - Inner join of user groups and client scopes produces effective user permissions
- Access token scope
 - Holds the effective permissions during a session

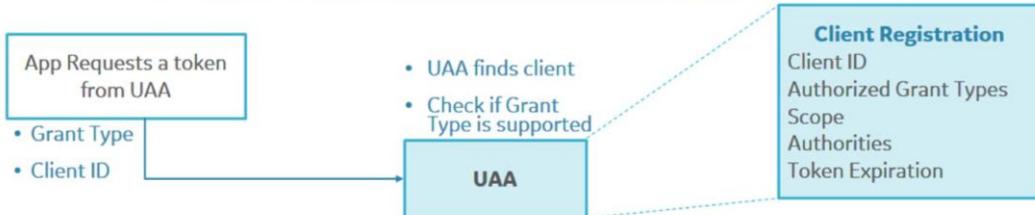


Client, as the application or service identity has a list of client authorities:

- List<String> of scopes that represents the permissions the client has by itself
- Scopes/permission field - represent the permissions that the client uses when acting on behalf of a user
- Authorities - used with client_credentials grant type



Requesting a Token



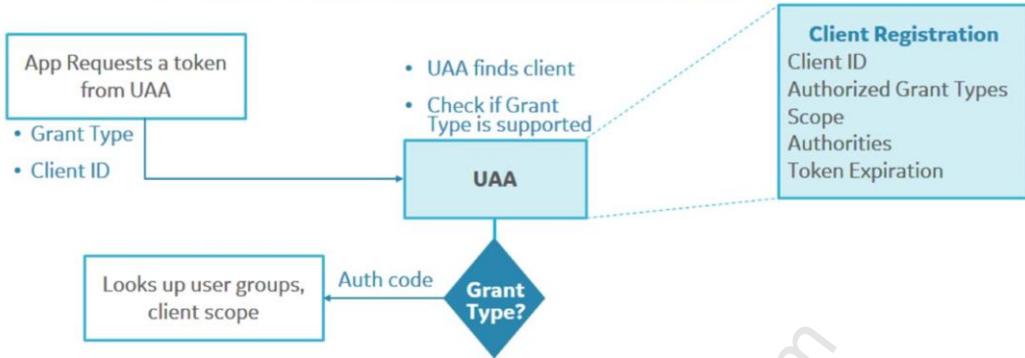
© 2016 General Electric Company - All rights reserved.

116

When an app requests a token, the required parameters are grant type and client ID. The UAA service checks the client registration to see if the grant type requested is supported by that client.



Requesting a Token



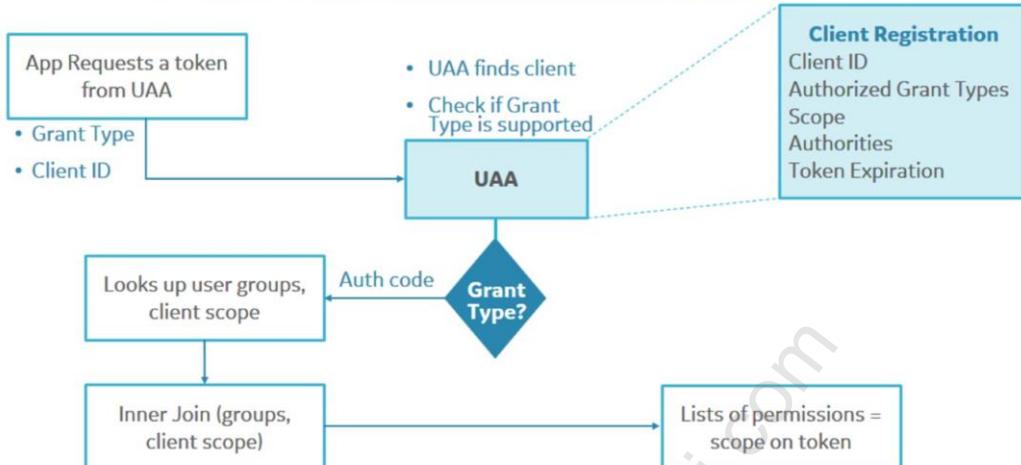
© 2016 General Electric Company - All rights reserved.

117

Depending on the grant type, UAA builds the list of permissions (scopes) for the token. If the grant type is authorization code, UAA looks up the client scopes and user groups to find the app permissions (on behalf of the user).



Requesting a Token

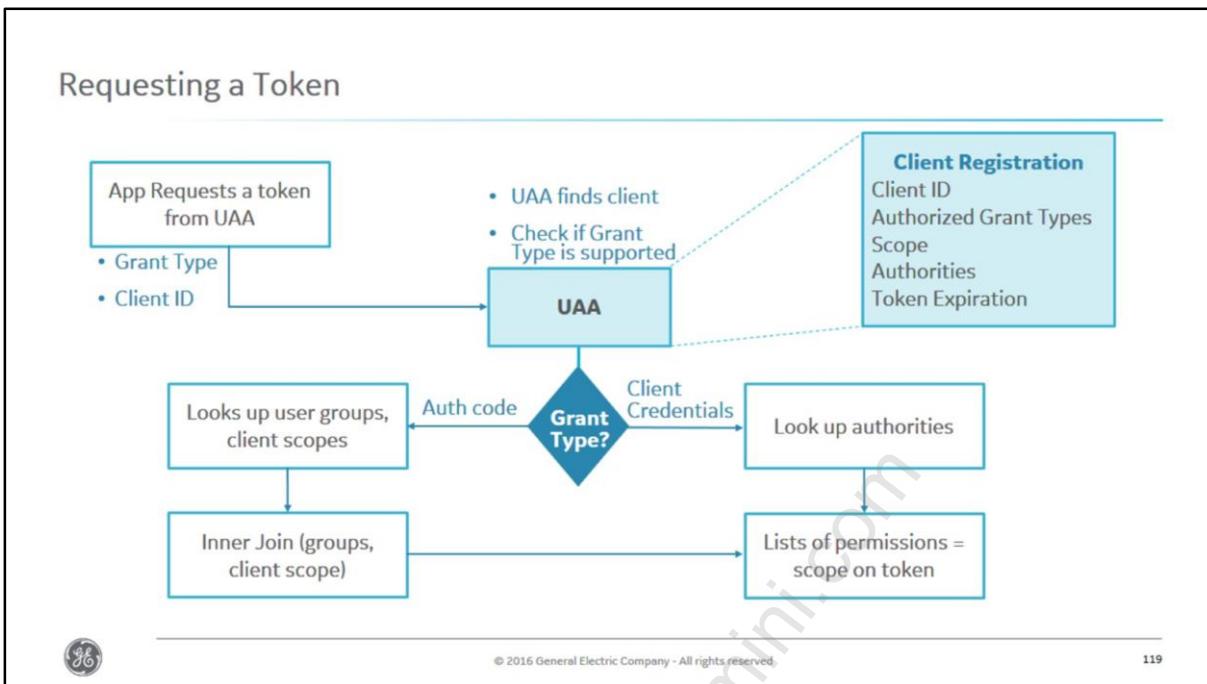


© 2016 General Electric Company - All rights reserved.

118

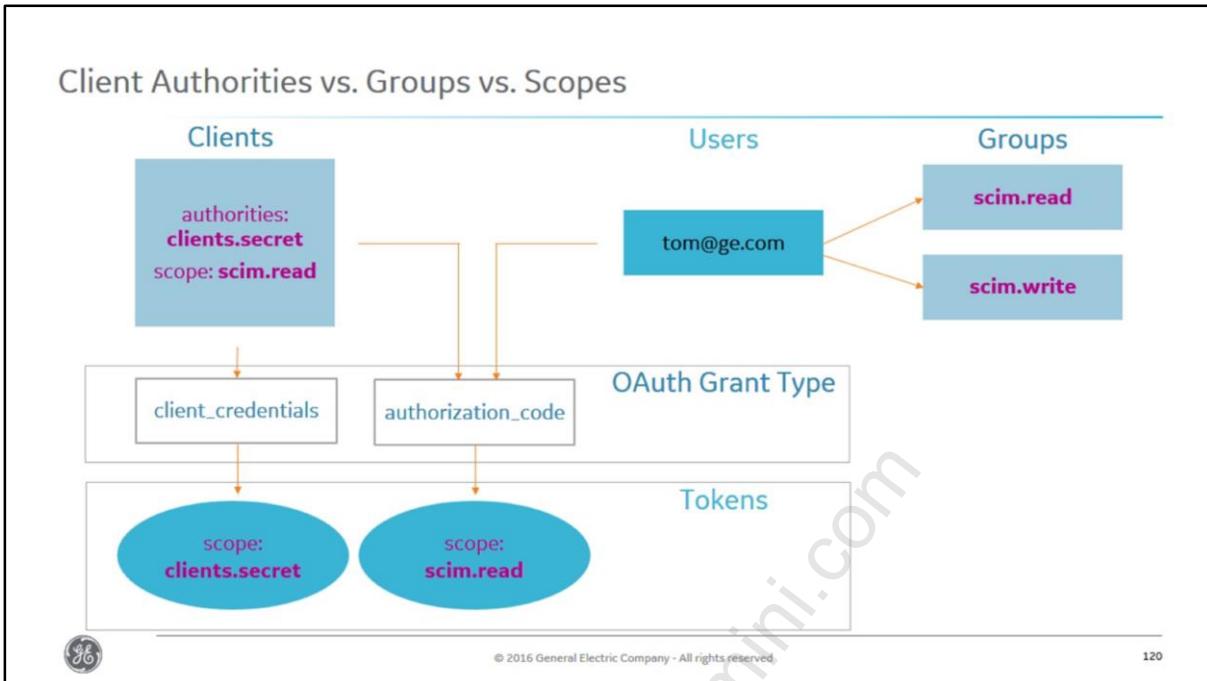
An inner join is done on the client scopes and the user permissions (groups) to get the final list of permissions, which are the scopes listed on the token.





If the grant type is client credentials, UAA just looks up the client authorities and uses the authorities in the list of permissions which are the scopes on the token.



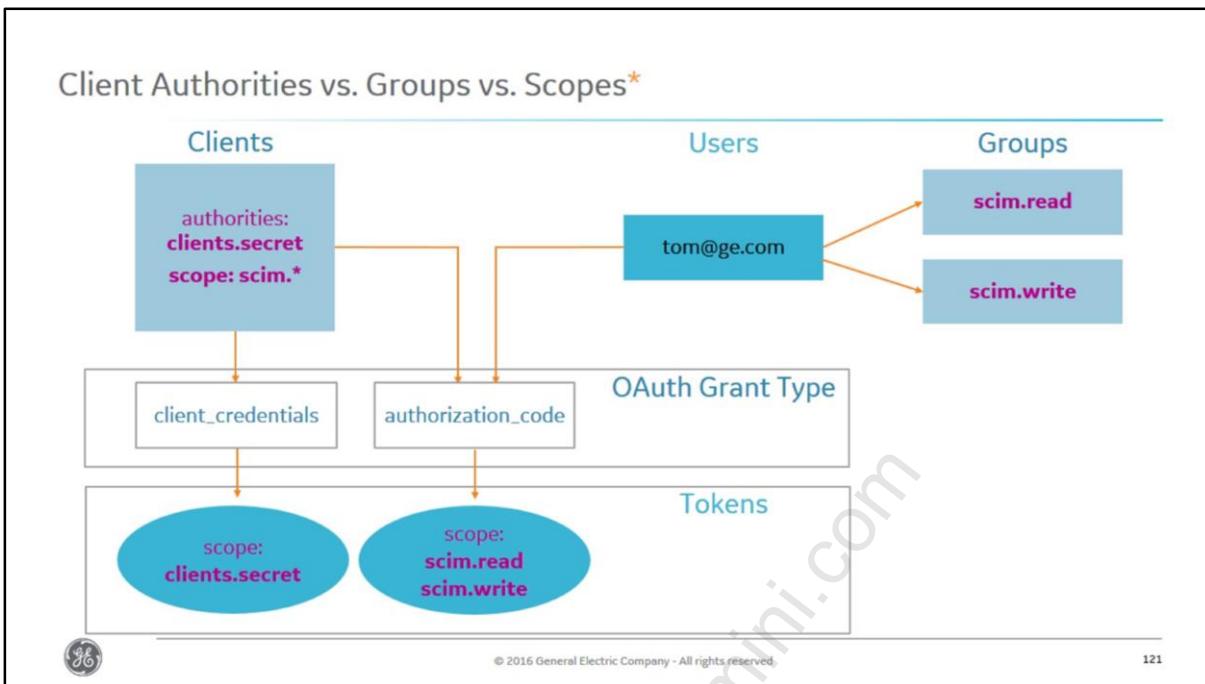


In the case of authorization code grant type user **tom@ge.com** is member of scim.read and scim.write groups. The client scope has scim.read in it. The effective scope shows up in the token, which is scim.read.

In the process of client credentials grant type the effective set of permissions that the token will have are limited to the client authorities.

This restricts and compartmentalize the level of access any individual application may have, so if you are building an administrative application you will allow users to access to more sensitive data or perform more sensitive tasks.

On the other hand, if you are building a simple data sharing application you may restrict the capabilities or privilege set of the users to match the context of the application.



In this example the client scope has a wild card allowing user **tom@ge.com** to retrieve a token, which provides him with scim.read and scim.write.

Create Groups

```
File Edit View Search Terminal Help  
Syntax: uaac group add scim.read  
Syntax: uaac group add scim.write  
Permission
```



© 2016 General Electric Company - All rights reserved.

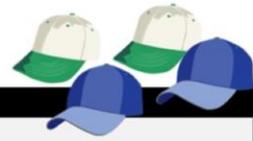
122

Next, create the groups for required permissions

```
uaac group add scim.read  
uaac group add scim.write
```



Add Users to Groups



```
File Edit View Search Terminal Help  
Syntax: uaac member add scim.read <user name>  
          ^             ^  
          |             |  
Assigned   user name  
membership to  
required scope
```



© 2016 General Electric Company - All rights reserved.

123

Then assign membership to the required scopes:

```
uaac member add scim.read <user_name>
```



Security

Lab 4: Security

</Lab>

- Exercise 3: Adding a client and users to UAA



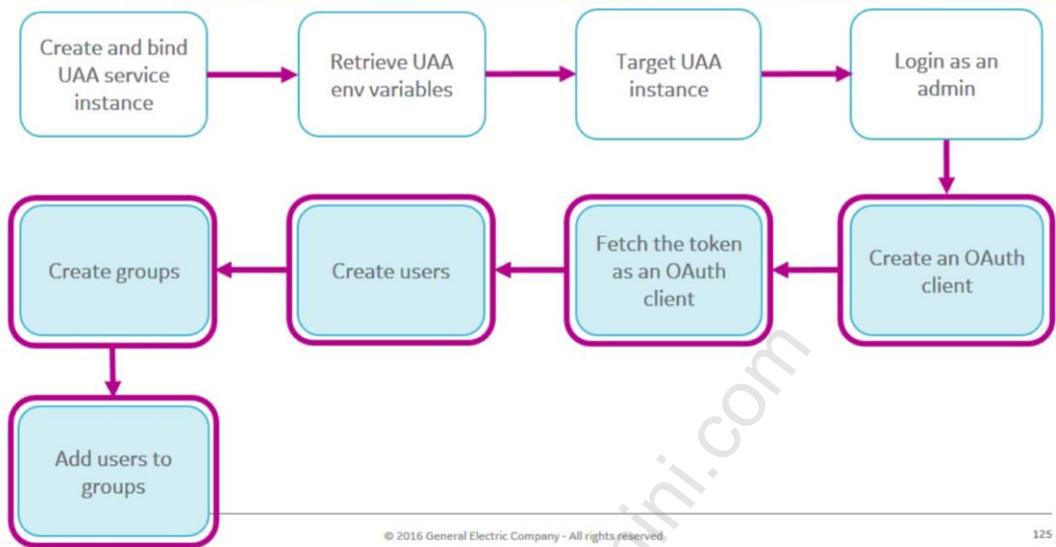
© 2016 General Electric Company - All rights reserved.

124

Exercise 3: In this exercise you will create an OAuth2 client you use to create users for in your UAA instance. When you create a UAA service instance, a default administrator account (admin client) is automatically generated that contains *all* permissions. As a best practice, you create an OAuth2 client and define the scopes instead of using the admin client to create users.



Security Lab Process Flow



Best Practices

- Read the [documentation](#)
- Use client credentials grant for devices
 - Give each device a client id and secret
 - Use [JWT Bearer Profile](#) for certificate-based authentication



© 2016 General Electric Company - All rights reserved.

126



Security

Access Control Service (ACS)



© 2016 General Electric Company - All rights reserved.

127



Access Control Service

Requires UAA

Services offered

- Attribute management
- Policy management
- Policy evaluation

Access Control Service

Use this service to provide a more powerful framework than the basic User Account and Authentication.

PREDIX

Features

- Ability to maintain access-decision data as policies and attributes
- Support for fine-grained authorization policies



© 2016 General Electric Company - All rights reserved.

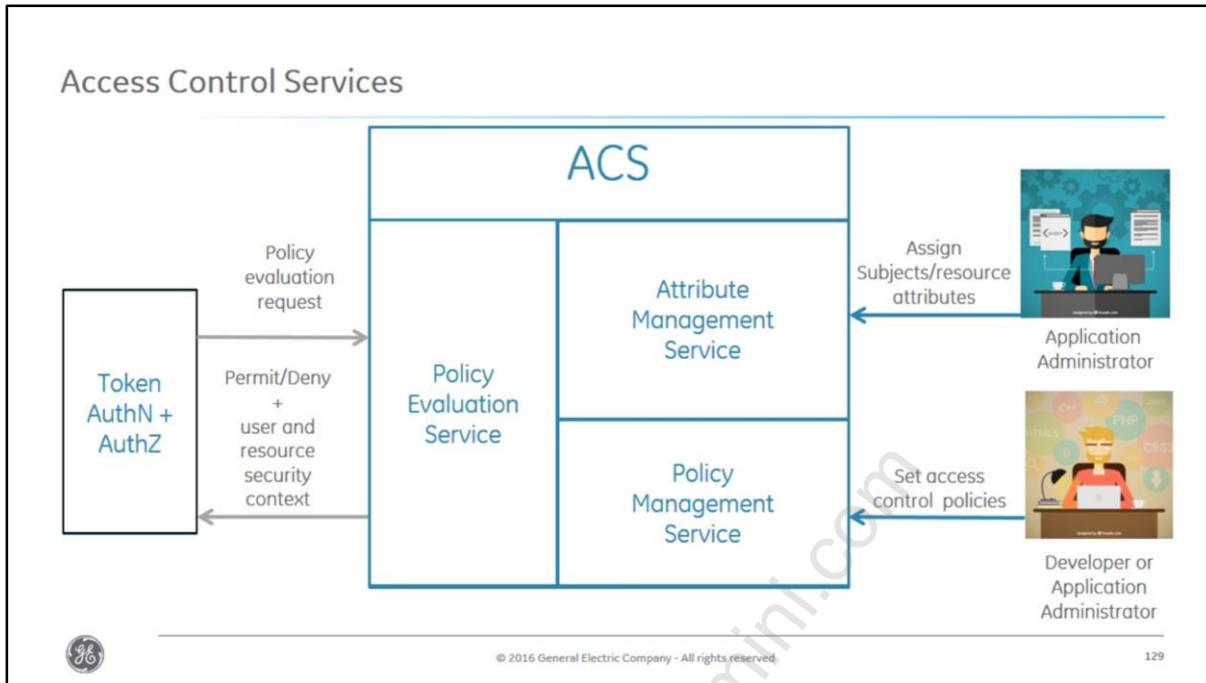
128

To use ACS, UAA must first be configured. ACS adds three services: policy evaluation, policy management, and attribute management.

Features:

- Ability to maintain access-decision data as policies and attributes
- Support for fine-grained authorization policies





The Predix platform provides ACS for application developers to add granular authorization mechanisms to access web applications and services. ACS works in conjunction with the User Account and Authentication (UAA) service in Cloud Foundry.

The high-level process flow for the ACS service is as follows:

- You first define your resources and subjects using the Attribute Management Service.
- Using the Policy Management service, you then define your policies (who has access to what) that are based on your attributes.
- At runtime, your client sends an evaluation request to the Policy Evaluation Service and gets back a PERMIT or DENY response plus any attributes discovered for the resource or subject.

Access Control Services consist of the following primary services:

The **Policy Management service** allows you (with required permissions) to create, read, update, and delete access-control policies. A security policy contains a set of rules that determine the required permissions for the specified subjects and resources. The rules can take into consideration the user attributes, the action the user wants to perform, the resource URI, and any resource attributes that further describe the resource.

The **Attribute Management service** allows you to create attributes for users and resources. Attributes are characteristics of a user or resource that can be used to make access-control decisions. An attribute is identified by an issuer, the entity that asserts the attribute, and a name that describes the attribute. Some examples of user and resource attributes include the organization, site, and group to which a resource belongs. Attributes are used in conjunction with access-control policies for user authorization.

The **Policy Evaluation service** is an internal ACS service that evaluates policies based on web service requests for authorization. A web service request is sent to the policy evaluation service, which then validates the request against a policy and defined attributes and returns a decision of permit or deny for the request. The request from a web service consists of the following components:

Resource information: Relative URI path of the request.

Subject information: The Policy Evaluation service extracts this information from the OAuth2 token used to authenticate with the web service.

Action information: HTTP method used for the request.



arun.moses@capgemini.com



ACS Features

Attribute Based Access Control (ABAC)

- Attribute store for
 - Subjects: entities that do things
 - Resources: entities that have things done to them
- Policy store
 - How subject and resource attributes combine to determine privileges
- Policy evaluation
 - Given a subject, action, and resource determine if operation is allowed



© 2016 General Electric Company - All rights reserved.

130

ACS was designed to address the access control limitations of OAuth. It is essentially a comprehensive solution for building a RESTful API that supports Attribute Based Access Control.

Attribute based access control means that every single identity involved in an Access Control decision has a set of attributes/properties such as 'I belong to this group,' 'I am a member of such organization', 'I have this license'. All these attributes are used to make Access Control decisions.

ACS is an Attribute Store for subjects and resources.

For instance; if users are members of a group, then the users can act on that resource (this is a policy) ACS also define rules and it's the ultimate decider, it holds attributes and policies and decides whether things are allowed or not allowed.



Attributes

Subject	Resource
<ul style="list-style-type: none">• tom@ge.com is an analyst• tom@ge.com is a member of the research group <p>subject identifier tom@ge.com attributes role: analyst group: researchers</p>	<ul style="list-style-type: none">• The asset with id 123 is located at the San Ramon site• The asset with id 123 belongs to users in the research group <p>resource identifier /asset123 attributes site: san-ramon group: researchers</p>



© 2016 General Electric Company - All rights reserved.

131

Attributes are characteristics of things (subjects or resources). Here are examples of a subject and a resource with their attributes.



Attribute Management Service

Allows you to create attributes for users and resources used to make access-control decisions

```

"subjectIdentifier": "supervisor",
  "attributes": [
    {
      "name": "role",
      "value": "ge_supervisor",
      "issuer": "https://acs.attributes.int"
    }
  ],
  "subjectId": "ge_supervisor"
}
  
```

"attributes" Specifies the attributes that the subject must have for the policy to be considered.

"name" optional information to identify a policy.



© 2016 General Electric Company - All rights reserved.

132

The Attribute Management service allows you to create attributes for users and resources. Attributes are characteristics of a user or resource that can be used to make access-control decisions. An attribute is identified by an issuer, the entity that asserts the attribute, and a name that describes the attribute. Some examples of user and resource attributes include the organization, site, and group to which a resource belongs. Attributes are used in conjunction with access-control policies for user authorization.

- Resources: an identifier like a role or group.
eg. {"resourceIdentifier": "customers"}
- Subjects: identity with name, value & issuer
eg. {"subjectIdentifier": "/subject/Acme Site Director", "attributes": [{"name": "role", "value": "Site_Director"}, {"issuer": "https://acs.attributes.int"}]}
- Policy Set: Leverages resources and subjects to issue a 'PERMIT' or 'DENY' to endpoints and actions.



Policy Management Service

- Allows you to create, read, update, and delete access-control policies
- Policies determine required permissions for subjects and resources

```

{
  "name": "Locomotive-ACS-policy",
  "policies": [
    {
      "name": "allow-all-HTTP-requests-for-supervisor",
      "target": {
        "resource": {
          "uriTemplate": "/asset123"
        },
        "subject": {
          "name": "has-role",
          "attributes": [
            {
              "issuer": "http://acs.attributes.int",
              "name": "role"
            }
          ]
        }
      },
      "conditions": [
        {
          "name": "",
          "condition": "match.single(subject.attributes ('https://acs.attributes.int', 'role'), 'ge_supervisor')"
        }
      ],
      "effect": "PERMIT"
    }
  ]
}
  
```

"uriTemplate" Specifies the URI template that the resource URI in the evaluation request must match.

"attributes" Specifies the attributes that the resources must have for the policy to be considered.

© 2016 General Electric Company - All rights reserved.

133

The Predix platform provides ACS for application developers to add granular authorization mechanisms to access web applications and services. ACS works in conjunction with the User Account and Authentication (UAA) service in Cloud Foundry.

Access Control Services consist of the following primary services:

The Policy Management service allows you (with required permissions) to create, read, update, and delete access-control policies. A security policy contains a set of rules that determine the required permissions for the specified subjects and resources. The rules can take into consideration the user attributes, the action the user wants to perform, the resource URI, and any resource attributes that further describe the resource.

Policy Set: Leverages resources and subjects to issue a 'PERMIT' or 'DENY' to endpoints and actions.



Policy Evaluation Process

- Client sends a request for authorization
 - Can a subject perform an action on a resource
- ACS performs
 - Attribute discovery
 - Policy evaluation
- Client receives
 - Authorization decision (permit | deny)
 - Discovered attributes

```
{
  "effect": "PERMIT",
  "subjectAttributes": [
    {
      "issuer": "https://acs.attributes.int",
      "name": "role",
      "value": "geuser"
    }
  ],
  "resourceAttributes": [
    {
      "issuer": "https://acs.attributes.int",
      "name": "locomotive",
      "value": "deltaqueen"
    }
  ],
  "resolvedResourceUris": [
    "asset123"
  ]
}
```



© 2016 General Electric Company - All rights reserved.

134

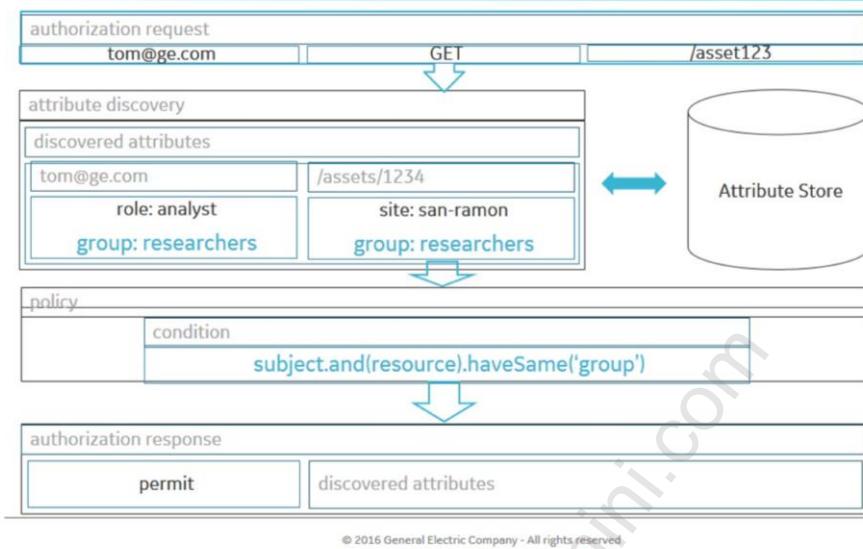
A REST API microservice receives a GET, DELETE, POST , POST, ADD request on a specific URI.

This request is sent to ACS which in turn discovers all the attributes of the user and all the attributes associated with the resource the user wants to act upon/change, and the actual action the user wants to perform.

ACS first performs an attribute discovery, and then a policy evaluation and it returns an authorization decision (permit | deny) response.



Policy Evaluation Process Flow Sample



When the authorization request is submitted, these are the discovered attributes for the subject ("tom@ge.com") and resource ("asset123").

In your policy you define a condition to specify when the policy is to be applied (when the subject and resource have the same group), and if the condition is met, the response to be returned (PERMIT).



Policy Evaluation Service

- Determines whether a subject is allowed the requested action on the specified resource based on its attributes.
- Sample policy evaluation request:
Subject **ge_supervisor** requesting **GET** action on resource **asset123**

The screenshot shows a JSON object with the following structure:

```
{
  "action": "GET",
  "resourceIdentifier": "/asset123",
  "subjectAttributes": [
    {
      "issuer": "https://acs.attributes.int",
      "name": "role",
      "scopes": [
        "ge_supervisor"
      ]
    }
  ]
}
```

Annotations explain the fields:

- "action"**: specifies the action (in the RESTful endpoint) that the policy permits for a specified resource. If you do not specify an action, all operations are allowed.
- "name"**: optional information to identify a policy.



© 2016 General Electric Company - All rights reserved.

136

The Policy Evaluation service is an internal ACS service that evaluates policies based on web service requests for authorization. A web service request is sent to the policy evaluation service, which then validates the request against a policy and defined attributes and returns a decision of permit or deny for the request. The request from a web service consists of the following components:

- Resource information: Relative URI path of the request.
- Subject information: The Policy Evaluation service extracts this information from the OAuth2 token used to authenticate with the web service.
- Action information: HTTP method used for the request.
- "effect" Specifies the access-control decision the Policy Evaluation service returns when the policy target matches a request, and all policy conditions return true. The effect can be either DENY or PERMIT.
- "condition" Specifies a predicate that must be satisfied for a rule to be assigned its effect. This element is optional. For example, the following condition matches the records-viewer role:
- You can use the following evaluation methods: match.single and match.any



Policy Evaluation Service

- Determines whether a subject is allowed the requested action on the specified resource based on its attributes.
- Sample policy evaluation response:
Subject **ge_supervisor** is permitted to GET resource **asset123**

```
{
  "effect": "PERMIT",
  "subjectAttributes": [
    {
      "issuer": "https://acs.attributes.int",
      "name": "role",
      "value": "ge_supervisor"
    }
  ],
  "resolvedResourceUris": [
    "asset123"
  ]
}
```

"effect" specifies the access control decision returned by the policy evaluation service. It can be PERMIT or DENY

© 2016 General Electric Company - All rights reserved.

137

The Policy Evaluation service is an internal ACS service that evaluates policies based on web service requests for authorization. A web service request is sent to the policy evaluation service, which then validates the request against a policy and defined attributes and returns a decision of permit or deny for the request. The request from a web service consists of the following components:

- Resource information: Relative URI path of the request.
- Subject information: The Policy Evaluation service extracts this information from the OAuth2 token used to authenticate with the web service.
- Action information: HTTP method used for the request.
- "effect" Specifies the access-control decision the Policy Evaluation service returns when the policy target matches a request, and all policy conditions return true. The effect can be either DENY or PERMIT.
- "condition" Specifies a predicate that must be satisfied for a rule to be assigned its effect. This element is optional. For example, the following condition matches the records-viewer role:"
- You can use the following evaluation methods: match.single and match.any



Create an ACS Instance

File Edit View Search Terminal Help

Syntax: `cf create-service <ACS service name> <Plan>`
`<acs service instance name> -c '{"trustedIssuers":["<uaa_instance_issuerID>"]}'`

```
{predix@localhost ~}$ cf create-service predix-acs-training Free  
acs service instance name -c '{"trustedIssuers":["https://<UAA Zone ID>.predix-uaa-  
training.run.aws-usw02-pr.ice.predix.io/oauth/token"]}'
```



© 2016 General Electric Company - All rights reserved.

138

Before you begin, ensure an instance of the UAA service has been configured as your trusted issuer. Then you can create a predix-acs-training service instance. Use cf marketplace to view the elements.

```
cf create-service predix-acs-training <plan> <my_acs_instance> -c  
'{"trustedIssuers":["<uaa_instance_issuerID>"]}'
```

Replace <uaa_instance_issuer_ID> with the issuer ID (not the uri) of your UAA instance.



Locating ACS Environment Variables

```
File Edit View Search Terminal Help
"Credentials": {
    "uri": "http://predix-acs-training.run.aws-usw02-pr.ice.predix.io",
    "zone": {
        "http-header-name": "Predix-Zone-Id",
        "http-header-value": "b3d0d15a-71fd-423c-9316-257b991fe7e4"
        "oauth-scope": "predix-acs-training.zones.b3d0d15a-71fd-423c-9316-257b991fe7e4.user".
    }
},
"label": "predix-acs-training",
"name": "training_acs_instance",
"plan": "Free",
"tags": []
}
```



© 2016 General Electric Company - All rights reserved.

139

You must bind your application to your ACS instance to provision its connection details in the VCAP_SERVICES environment variable. Cloud Foundry runtime uses the VCAP_SERVICES environmental variable to communicate with a deployed application about its environment.

2. Bind your application to the ACS instance. Execute the bind-service command:

```
cf bind-service <your_app_name> <acs_instance_name>
```

Verify the ACS instance connection details from VCAP_SERVICES

```
cf env <your_app_name>
```



Client Update Command

```
File Edit View Search Terminal Help

{predix@localhost ~}$ uaac client update --authorities
"clients.read clients.write scim.read scim.write" ← Original authorities
"acs.policies.read acs.policies.write acs.attributes.read acs.attributes.write" ← ACS additions
"Predix-acs-training.zones.ccff1702-ef11-4769-a527-deade4c917b0.user"
}
Oauth-scope environmental
variable from cf env <app>


© 2016 General Electric Company - All rights reserved.
140
```

When you update the client for ACS permissions, don't forget to include the original set of authorities you established when you created the client. Add to that list the **zones.<xxx>.admin** variable from admin authorities, which you can see from the **uaac clients** command. Add the necessary ACS policies and attributes authorities.

And finally, include the ACS oauth-scope environmental variable **predix-acs-training.zones.xxx.user** from your application (**cf env <application name>**).



Security

Lab 4: Security

</Lab>

- **Exercise 4:** Create an ACS instance
- **Exercise 5:** Bind your application to the ACS instance
- **Exercise 6:** Update an OAuth2 client to work with ACS
- **Exercise 7:** Add and Evaluate an ACS policy



© 2016 General Electric Company - All rights reserved.

141

Exercise 4: In this exercise you will create an ACS instance, and bind your application to the instance.

Exercise 5: You must bind your application to your ACS instance to provision its connection details in the VCAP_SERVICES environment variable.

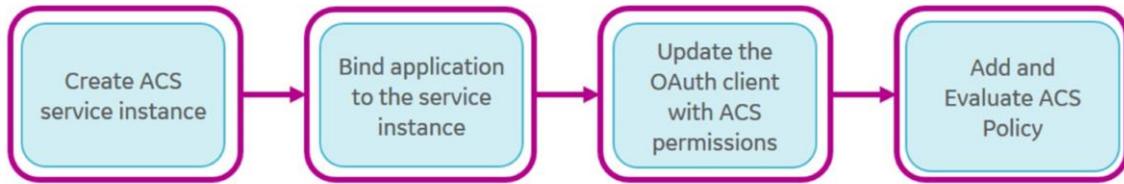
Exercise 6: To enable applications to request authorization or policy evaluation from ACS, you need to update your OAuth2 client with the required ACS scopes and authorities. In this exercise, you will establish your OAuth2 client to handle tokens sent by the application or client

Exercise 7: To enable applications to manage policies and attributes using ACS, you need to update your OAuth2 client with the required OAuth2 scopes and authorities. In this exercise, you will establish your OAuth2 client to handle Policy Management and Attribute Management Services. This will handle tokens sent by the application or client.



Predix Application Security –Setup for ACS Authorization

You must have a UAA instance to use ACS, as the two work together



© 2016 General Electric Company - All rights reserved.

142

Go to [predix.io/ACS/documentation/Using Access Control Services](https://predix.io/ACS/documentation/Using%20Access%20Control%20Services)
<https://www.predix.io/docs/#NBx9h4GJ> to show creation of attributes and policies.



Module Summary: Security

- UAA architecture details how a user is authorized to access data
- UAA is used to authenticate application users
- OAuth2 client represents an app, including scopes and authorities
- Authorization code & client credentials are the most used grant types
- ACS uses policies and attributes to make access control decisions



© 2016 General Electric Company - All rights reserved.

143



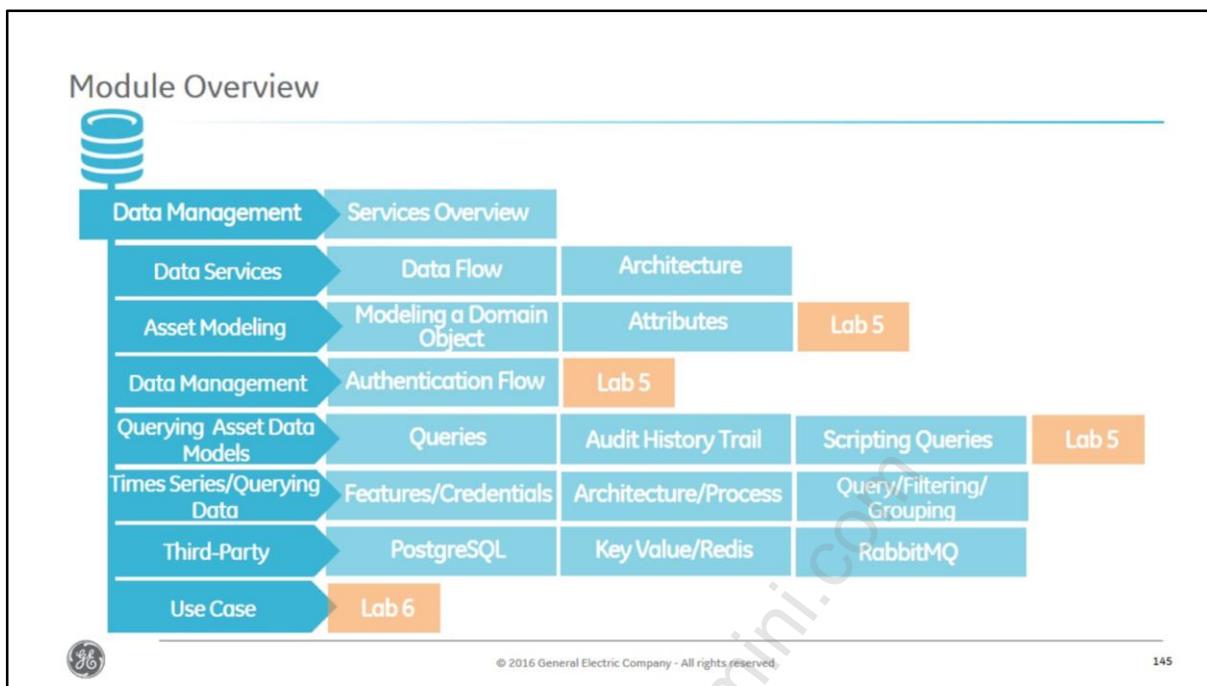
Data Management



© 2016 General Electric Company - All rights reserved.

144





Module Objectives

By the end of this module you will be able to:

- Describe Predix Data Management Services
- Describe Data Process Flow and Architecture
- Upload asset data into the Predix asset service
- Query assets data and return specific information about an asset
- Describe Time Series Architecture, Components and Data Flow

Completion of the following lab activities will reinforce the concepts covered in this module:

- Lab 5: Using the Asset Service
- Lab 6: Locomotive Client Application: Use Case



© 2016 General Electric Company - All rights reserved.

146



Predix Data Management Services

Predix Asset	Enables App developers to create and store asset instances, properties & relationships
Predix Time Series	Manage, ingest, store, query data Real-time streaming data Group data points by tags, time ranges, or values Perform aggregate functions (avg, min, max, trends, interpolations)
Predix Blobstore	Store large byte arrays in cloud S3 API compatible Scalable, reliable, and high availability Multi-tenancy



© 2016 General Electric Company - All rights reserved.

147

Asset

The Predix Asset service provides REST APIs to support asset modeling. The Asset service enables application developers to create and store assets that define asset properties as well as relationships between assets and other modeling elements. Developers can store asset instance data.

Time Series

Time Series data is a sequence of data points collected at set time intervals over a continuous period of time. Sensor data is an example of a common way to generate time series data. A Time Series data store requires a measurement with a corresponding timestamp. The Time Series service provides an attributes field to include additional relevant details about that specific data point, such as units or site.

Blobstore

The Blobstore service provides a way to store large byte arrays in the cloud indefinitely.

Modeling Assets is the first step where you create and store machine asset models and instances.

Once we have data queried in a helpful way, then the analysis begins. Data Analytics provides the build-upload-validate-release-execute workflow using the Analytics catalog and Runtime services.



Third-Party Data Management Services

POSTGRESQL: SQL DB	Use PostgreSQL as an ODBMS to store data securely for retrieval at the request of other software apps
REDIS: Key-Value Store	Key-value store NoSQL Can contain strings, hashes, lists, sets, sorted sets, bitmaps, and HyperLogLog algorithms
RABBITMQ: Message Queue	Persistent, highly available, reliable messaging between apps, components, and devices



© 2016 General Electric Company - All rights reserved.

148

SQL Database

Use the PostgreSQL as a service object-relational database management system to store data securely for retrieval at the request of other software applications.

The database can handle workloads ranging from single-machine apps to internet-facing applications with many concurrent users.

Redis

Use this advanced key-value cache and store database for a lightweight and flexible way to work with data. This type of key-value storage model is also referred to as NoSQL. Keys can contain strings, hashes, lists, sets, sorted sets, bitmaps, and HyperLogLog algorithms. HyperLogLog – a class of algorithms that use randomization in order to provide an approximation of the number of unique elements in a set using just a constant, and small amount of memory. (Philippe Flajolet)

RabbitMQ

Use this message broker software service to provide persistent, highly available, reliable messaging between applications, components, and devices. The message queue service supports a variety of messaging protocols, many clients, and flexible routing with built-in direct, fanout, topic, and header exchange types.



Data Management

Asset Data Service



© 2016 General Electric Company - All rights reserved.

149



A Use Case for Asset Data Service

Sample Asset – Locomotive

Use Case: Locomotive has a down engine

- Where can I find an idle engine?
- Which fleet?
- Closest location?



© 2016 General Electric Company - All rights reserved.

150

Before we examine the Predix Asset Data Service and what it provides, we first need to answer the question; what is an asset model and why is it so important?

Let's take a look at a typical asset: a locomotive

A locomotive is comprised of thousands of parts: engines, wheels, cars, etc.. Each of those parts themselves can contain hundreds or thousands of other parts. So, our locomotive asset contains other assets. Additionally, these assets can have a relationship with each other. A locomotive can belong to a fleet of locomotives. That fleet is also an asset.

Let's say we're a transportation company and one of our locomotive's engine goes down. We need to quickly locate a replacement engine that is either an exact match or one of the approved engine types defined for that locomotive model. Now imagine that we own thousands of locomotives located in fleets dispersed across the country.

Or, during a scheduled maintenance check, it is determined that the engine installed in the locomotive is older than the approved age for that engine type as defined in the asset model. This could trigger an alert to replace the engine before any problems set in (degradation in performance, breakdowns, etc.)

Before you can begin collecting and analyzing data from your assets, you need to first construct a model or blueprint of your assets to set threshold values for attributes/properties so they can be evaluated against data collected from sensors on your assets.



Asset Data Service

- Provides REST APIs to support asset modeling
- Create, update and store asset model data
- Create asset instance representations
- Quickly retrieve vast amounts of asset data

Asset Data

Create and store machine asset models and instances.

PREDIX



© 2016 General Electric Company - All rights reserved.

151

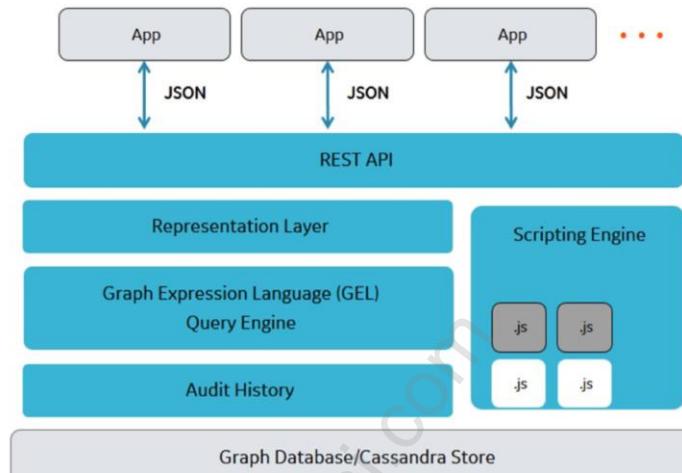
The Predix Asset Service provides REST APIs that support asset modeling. The Asset service enables application developers to create and store asset models that describe machine types, as well as create instance representations for those machines. You already know what your asset model looks like; the Asset Data service lets you bring your asset model into the cloud. Developers can create custom modeling elements that meet their unique domain needs.

With the Predix Asset service, an application developer can create an asset model that describes the logical component structure of all locomotives in an organization, and then create instances of that model to represent each individual locomotive in an organization.



Architecture

- REST API layer
- Representation Layer
- Query Engine
- Database
- Scripting Engine



© 2016 General Electric Company - All rights reserved.

152

Let's take a look at the internal workings of the Asset Data service.

The Asset Data service consists of the following components:

- A REST API layer that exposes REST endpoints to allow your client application to interact with the Asset model. Here, application developers enter JSON that describes their asset objects. You use REST APIs to create and update your asset model
- A Representation layer that converts JSON data to RDF triples and back to JSON
- The Query engine enables developers to use JSON and the Graph Expression Language (GEL) to retrieve data about any object or any object property in the asset service data store
- Cassandra database - The Asset service back-end data store is a database that stores data as RDF triples. This makes querying datasets easier and faster
- The scripting engine is a service that allows users to bind their customer business logic to the Predix Asset Restful API



Data Management

Asset Modeling



© 2016 General Electric Company - All rights reserved.

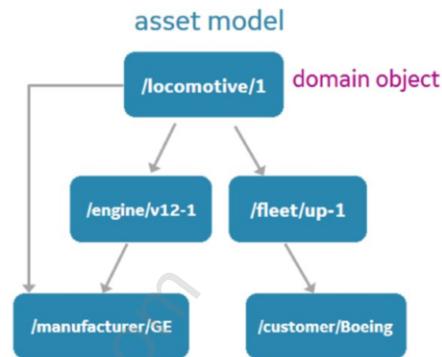
153



Asset Model

Represents information about your assets

- How they are configured
- How they are related
- ‘Suggested’ vs ‘actual’
- Supports custom models
- Hierarchical, network and other structures
- Represent modeling elements & asset instances
- Domain objects can be related to each other



© 2016 General Electric Company - All rights reserved.

154

An asset model represents the information that you are storing about your assets: how your assets are organized, and how they are related. Asset modeling refers to metadata about an asset – not the data sent by an asset's sensors. Asset models are “blue prints” for what your assets should look like: used to set threshold values for assets or define acceptable parameters of operation (max speed or max temperature for an engine, the different engine types that are acceptable for a given locomotive model, max number of days in between maintenance checks, etc.).

The Asset service APIs allow you to build a data model – a blueprint of what your asset objects should look like. This blueprint not only defines the metadata for your assets, but it is also used to define ‘acceptable parameters of operation’ or thresholds for your asset instances (e.g. maximum temp for an engine, approved engine types used in a specific locomotive model, etc.).

“Actual” or real-time data collected from sensors on your assets can be mapped and compared to the **“suggested”** thresholds and parameters defined in the asset model data to determine if there is a mismatch that could indicate problems or anomalies.

You design your asset model separately (outside of Predix) and then implement the model using the Asset Service APIs provided. You can define your assets using any model that meets your needs, for example, a hierarchical structure with parent asset and one or many peers and children. The Predix Asset service is highly flexible and supports hierarchical, network and other model structures.

Domain objects represent the modeling elements as well as the assets themselves. Domain objects are just entities that you can create and track/manage. Examples: Assets, Classifications, Part Numbers, Templates, Alerts, Customers, Events. A domain object is denoted by a forward slash and can have any number of properties or attributes, such as manufacturer name, an engine type, or a serial number.

Within an asset model, domain objects can be related to each other. For example, a locomotive asset has an engine asset, or a fleet asset is associated with a customer asset. The arrows here represent the relationship between domain objects.



Data Management

Using the Asset Service



© 2016 General Electric Company - All rights reserved.

155

Now let's take a look at how you use the Asset Service API's to load asset model data to the cloud.



Asset Service Functionality and Documentation

Allows you to view and update asset model

- REST operations:
 - GET, POST, PUT, DELETE, PATCH
- Query asset data:
 - Scripting
 - Auditing
 - Triggers

API Documentation

ASSET

- ▶ Audit Records
- ▶ Audit Snapshots
- ▶ Data Validation
- ▶ Scripts
- ▶ System Messages
- ▶ Tenant Configurations
- ▶ Triggers



© 2016 General Electric Company - All rights reserved.

156

In the Predix catalog, clicking on the **Asset Data** tile displays the Asset API documentation

Client applications can access asset data using Asset service REST API endpoints. These endpoints provide a JSON interface where you POST the data that describes all of your assets. To use the APIs, your application makes HTTPS requests and parses the response.

The Asset Service is used to store and query asset model data. The service APIs enable you to create, update and delete asset data in your asset model.

The following CRUD operations are supported:

POST – add one or more asset entities

GET – retrieve one or more asset entities

PUT – update an existing entity (must specify all attributes for the entity)

DELETE – remove a single asset entity (cannot be used to delete all asset entities)

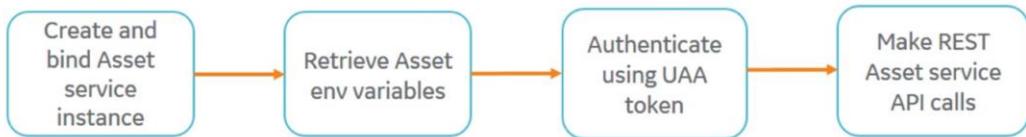
PATCH – allows you to update only selected properties for an asset instance as opposed to the PUT operation that forces you to update ALL properties for an instance. PATCH is more efficient – decreases the payload.

In addition to the APIs, the Asset service has a Graph Expression Language or GEL, that allows you to create custom filters when querying asset data. Rather than returning all asset entities, you narrow the size of a result set by adding filters to HTTP requests. Other query parameters can be used to specify specific attributes to be displayed or to limit the number of objects returned from a collection.



Using the Asset Service APIs

Prerequisite: Design Asset Model Schema



© 2016 General Electric Company - All rights reserved.

157

The are the high-level steps for using the Asset Data service.

Before you begin, you should already have your asset model schema defined.

To use the Asset Data service available in the Cloud Foundry marketplace, you need to create an instance of the asset service. This is done by running the Cloud Foundry command: **cf create-service** followed by the required parameters

The next step is to bind your client application to the asset service instance. This can be done by running the Cloud Foundry command: **cf bind-service** followed by the required parameters

To access and update asset model data, client applications must authenticate to the asset service. This requires getting a token from a UAA service.

User Account and Authentication, or UAA, is a web service provided by Cloud Foundry to manage users and OAuth2 clients. Its primary role is as an OAuth2 provider, issuing tokens for client applications to use when they act on behalf of Cloud Foundry users.

Once authenticated, your client application can make REST calls to the asset service to add, update and delete asset model data.



Data Management

Creating and Securing the Asset Service



© 2016 General Electric Company - All rights reserved.

158

Organizations do not want to have their asset data exposed to just anyone.

User Account and Authentication (UAA) is a web service provided by Cloud Foundry to manage users and OAuth2 clients. Its primary role is to issue tokens for client applications to use when they act on behalf of Cloud Foundry users. In collaboration with the login server, it can authenticate users with their Cloud Foundry credentials, and can act as an SSO service using those credentials (or others). The service provides endpoints for managing user accounts and for registering OAuth2 clients.



Authentication Flow Diagram



© 2016 General Electric Company - All rights reserved.

159

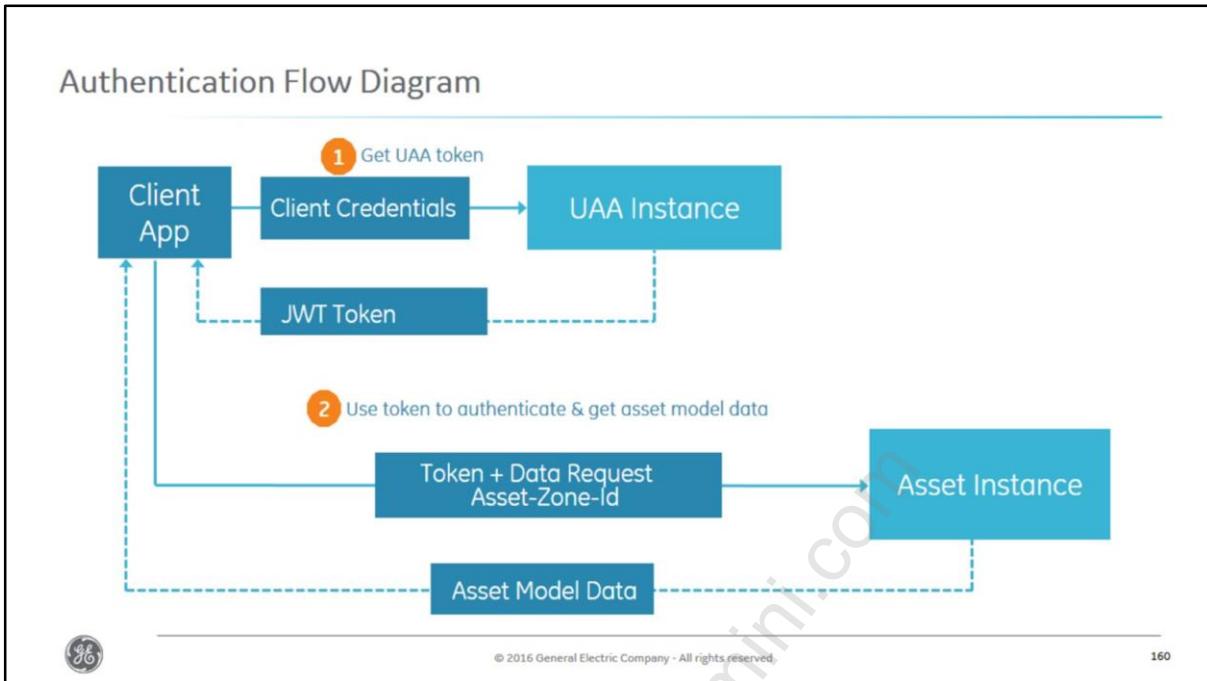
UAA stands for: [User Account and Authentication Service](#)

- Web applications need to be able to authenticate users and access platform resources on their behalf without collecting user credentials.
- UAA provides a single sign-on and delegates authorization to web applications
- Enables a Client web application, to act on behalf of a User, but with the User's permission

The authentication process can be broken down into 2 main steps:

1. Get UAA token – the client application submits an HTTP request to a UAA **service instance** and passes user credentials and a tenant ID. The UAA instance sends a response that includes a bearer token (JWT stands for Jason Web token)
2. Use token to authenticate & get asset model data – the client application submits a request for asset data to an asset service instance. The request includes the bearer token, a tenantID and the asset data to be retrieved or added. The Asset service instance returns a response that includes asset data



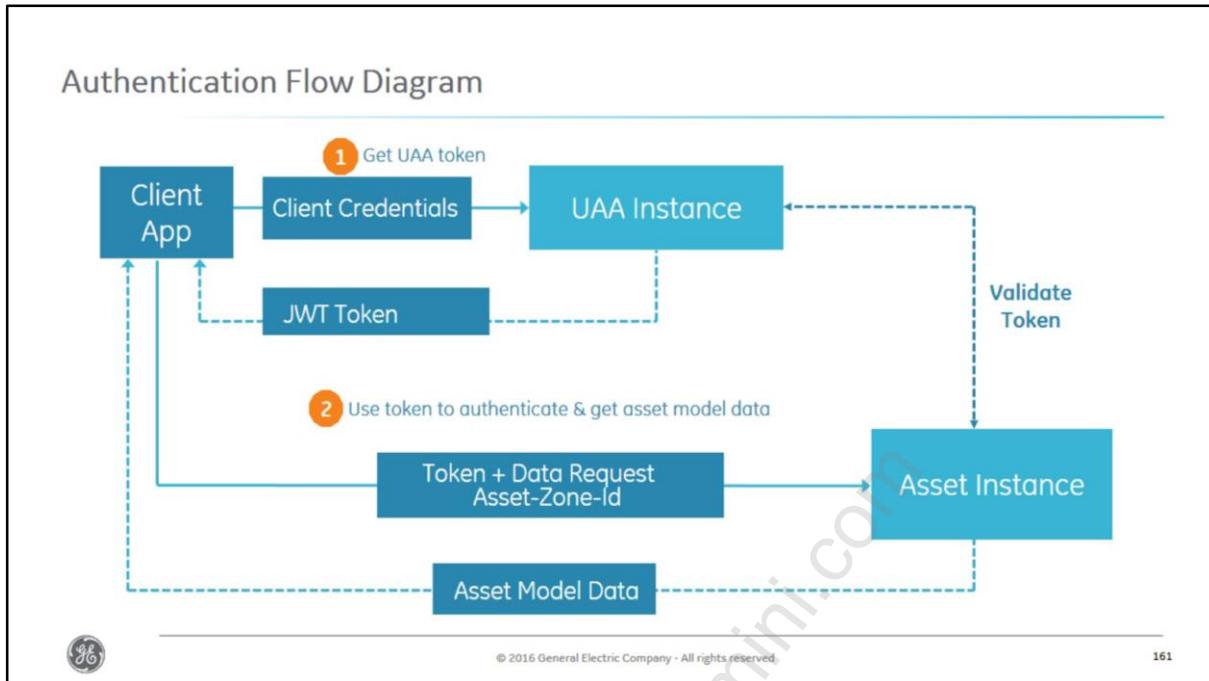


UAA stands for: [User Account and Authentication Service](#)

- Web applications need to be able to authenticate users and access platform resources on their behalf without collecting user credentials.
- UAA provides a single sign-on and delegates authorization to web applications
- Enables a Client web application, to act on behalf of a User, but with the User's permission

The authentication process can be broken down into 2 main steps:

1. Get UAA token – the client application submits an HTTP request to a UAA **service instance** and passes user credentials and a tenant ID. The UAA instance sends a response that includes a bearer token (JWT stands for Jason Web token)
2. Use token to authenticate & get asset model data – the client application submits a request for asset data to an asset service instance. The request includes the bearer token, a tenantID and the asset data to be retrieved or added. The Asset service instance returns a response that includes asset data



UAA stands for: [User Account and Authentication Service](#)

- Web applications need to be able to authenticate users and access platform resources on their behalf without collecting user credentials.
- UAA provides a single sign-on and delegates authorization to web applications
- Enables a Client web application, to act on behalf of a User, but with the User's permission

The authentication process can be broken down into 2 main steps:

1. Get UAA token – the client application submits an HTTP request to a UAA **service instance** and passes user credentials and a tenant ID. The UAA instance sends a response that includes a bearer token (JWT stands for Jason Web token)
2. Use token to authenticate & get asset model data – the client application submits a request for asset data to an asset service instance. The request includes the bearer token, a tenantID and the asset data to be retrieved or added. The Asset service instance returns a response that includes asset data



Getting a UAA Token

Use REST Client to get token from UAA endpoint

Requires:

- UAA issuerID
- Headers
 - Client credentials
 - Content type
- Body
 - grant_type

The screenshot shows a REST Client interface with a 'Get Token' tab and a 'GetUAAToken' sub-tab. The method is set to 'POST' and the URL is '<uaa_trusted_issuer_ID>'. The 'Body' tab is active, showing a key-value pair for 'grant_type' set to 'client_credentials'. Other tabs like 'Headers' and 'Params' are visible but not selected.



© 2016 General Electric Company - All rights reserved.

162

A client application retrieves a bearer token with a POST request. In our classroom environment, we use the REST Client to submit requests for asset data.

A request uses the POST API method with a UAA URL retrieved from the VCAP_SERVICES variable found in the output of the `cf env` command.

A POST request requires three headers:

- **Basic Authorization** that contains your UAA login credentials
- Content-type header of **x-www-form-urlencoded** that specifies we'll be sending data formatted in key-value pairs
- **X-tenant** header that specifies a tenantID (a tenant is a group of users that typically belong to an organization who share a common access with specific privileges)

A Content-type header indicates what type of content we're sending which allows the server to interpret the response correctly.

In the **body payload** specify a response type, grant type: `grant_type=client_credentials`

The UAA server returns a bearer token in the response. You use the token to retrieve asset data from your asset service instance.



Data Management

Lab 5: Using the Asset Service

</Lab>

- Exercise 1: Create Asset Service Instance
- Exercise 2: Configure UAA Client for Asset Service
- Exercise 3: Fetch a token for the UAA App Client



© 2016 General Electric Company - All rights reserved.

163

- Configure UAA Instance authorities
- Generate a token using postman

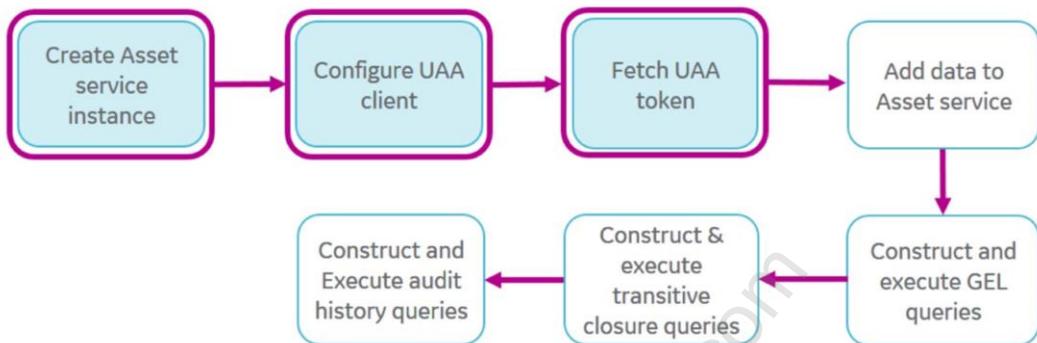
Exercise 1: To use a Predix Service, you must first create an instance of the service in Cloud Foundry. You will create your Asset service instance using the UAA service instance you created in the previous exercise; you must have the base URLs for UAA instances that the Asset service instance will trust.

Exercise 2: When you created your UAA client instance, an "admin" client was automatically generated for you. In this exercise, you provide your UAA client with the authority to write to the Asset database using the UAA command line interface (uaac).

Exercise 3: To access and update asset model data, users require a UAA token. Here, you use the REST client to request the token from the UAA Service. The token needs to be added to every data request to the Asset Service.



Asset Service Instance Lab Process Flow

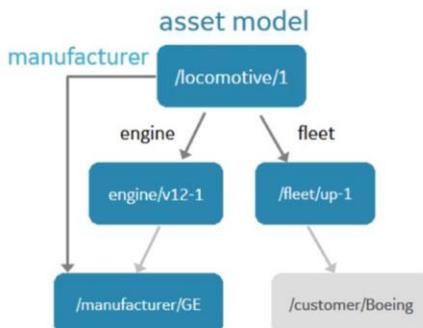


© 2016 General Electric Company - All rights reserved.

164



Modeling a Domain Object Using JSON



corresponding JSON

```
[
  {
    "uri": "/locomotive/1",
    "type": "Diesel-electric",
    "model": "ES44AC",
    "serial_no": "001",
    "emission_tier": "0+",
    "fleet": "/fleet/up-1",
    "manufacturer": "/manufacturer/GE",
    "engine": "/engine/v12-1",
    "hqLatLng": {"lat": 33.914605, "lng": -117.253374}
  }
]
```



© 2016 General Electric Company - All rights reserved.

165

You use JSON to define your asset model - the domain objects, their attributes and relationships between domain objects.

You also use JSON to create **instance** representations of your assets. Each piece of physical equipment is represented by an asset instance.

Properties defined for a domain object are **attributes**; even those properties that contain a uri (for example, "/manufacturer/GE") reference another domain object.

Here we have a sample asset model and the corresponding JSON that defines the model. The JSON is uploaded to the Asset service using the service APIs.

Note that the Customer domain object is not yet defined in the JSON for "locomotive 1" – with the flexible asset service, you can attributes to existing objects at any time using the APIs.

We also see that the domain object **customer** listed in the asset model in grey does not yet exist in the example JSON. With the flexible asset service it is easy to update attributes to existing objects at any time using the Asset API's



Defining an Asset Model

JSON defines domain objects

- Objects are unordered sets of name/value pairs
- URI uniquely identifies objects

Format

- Objects contained within braces
- Data separated by commas
- Strings require double-quotes

Well-formed JSON required

- Use JSONlint.com to validate

```
[ { "uri": "/locomotive/1", "type": "Diesel-electric", "model": "ES44AC", "serial_no": "001", "emission_tier": "0+", "fleet": "/fleet/up-1", "manufacturer": "/manufacturer/GE", "engine": "/engine/v12-1", "hqLatLng": { "lat": 33.914605, "lng": -117.253374 } } ]
```



© 2016 General Electric Company - All rights reserved.

166

Predix uses JSON to describe objects in an asset model. JSON stands for JavaScript Object Notation. It is a data-interchange format that is based on a subset of the JavaScript programming language. It is easy for humans to read and write, and is easy for machines to parse and generate.

JSON is built on two structures:

- A collection of name/value pairs.
- An unordered list of values (these can be stored as an array, vector, list, or sequence).

The Asset Data service requires a well-formed JSON with URI. The URI is like a primary key in that it uniquely identifies an asset instance stored in the database.

There are some syntax rules when working with JSON:

- **Objects are unordered sets of name/value pairs, meaning attributes do not need to be defined in any specific order** (example: name: value or "location": "/location/USA")
- **An object begins with a left curly brace { and ends with a right curly brace }**
- **Each attribute name is followed by a colon (colon) and the name/value pairs are separated by a comma (comma).**
- A value can be an array, object, string, number, boolean, null.
- **Names require double quotes**
- Strings can be URIs.
- **Data is separated by commas**
- Square brackets hold arrays

It's important to note that the asset service does not perform validation for JSON. You must validate your JSON yourself, using any of the available tools such as JSONlint.org

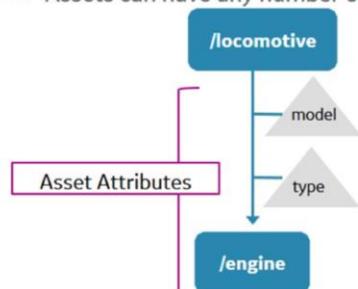
JSON.org is a useful website for grammar and learning.



Attributes

Provide more information about an asset

- Example: type:"Diesel-electric"
- Assign attributes at any level in the hierarchy
- Assets can have any number of attributes



© 2016 General Electric Company - All rights reserved.

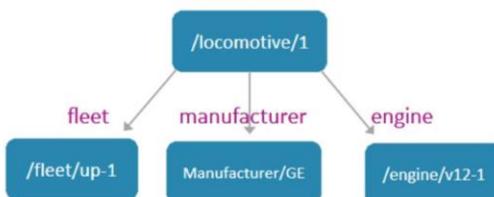
167

Customer-defined properties or **attributes** provide more information about an asset. Some attributes contain a simple value, such as a model name or an engine type. Other attributes may contain a pointer to other domain objects such as the manufacturer attribute which points to a GE manufacturer domain object. There is no limit to the number of attributes that can be assigned to a domain object.

Defining Relationships Between Assets

Assets can be associated with other assets

- Peer-to-peer and parent-child relationships
- Linked object URI
- Use JSON syntax: attribute: "linkedObjectURI"
 - Example: fleet: "/fleet/up-1"



```

[ {
  "uri": "/locomotive/1",
  "type": "Diesel-electric",
  "model": "ES44AC",
  "serial_no": "001",
  "emission_tier": "0+",
  "fleet": "/fleet/up-1",
  "manufacturer": "/manufacturer/GE",
  "engine": "/engine/v12-1",
  "hqLatLng": {"lat":33.914605, "lng": -117.253374}
}
]
  
```

A pink curly arrow points from the word "manufacturer" in the JSON code back to the "manufacturer" node in the diagram.



© 2016 General Electric Company - All rights reserved.

168

Within an asset model, assets can be associated with many other assets, whether in a network structure (peer-to-peer relationship) or as a parent-child relationship. This link or relationship is stored in an attribute, using the syntax "forward slash and the linked object's uri."

For example, this code snippet shows the JSON for the locomotive 1 asset. This locomotive asset is a member of the "up-1" fleet. Both assets are independent of each other, but they have a relationship that is defined in the **fleet** attribute of the locomotive asset. As we see here, the fleet attribute contains the URI of the up-1 fleet object.



Adding Asset Data

REST Client request

- Use **POST** API method
- Asset zoneID
- Add JSON to body

The screenshot shows two separate Postman requests. The top request is titled '<Asset_URI>' and has 'POST' selected, a URL of '<Asset_URI><Asset_endpoint>', and three checked headers: Content-Type, Predix, and Authorization. The bottom request is also titled '<Asset_URI><Asset_endpoint>' and has 'POST' selected, a URL of '<Asset_URI><Asset_endpoint>', and three checked headers: Authorization, Headers (3), and Body. The 'Body' tab is selected, and the content type is set to 'raw'. The JSON payload is:

```

1  [
2   {
3     "uri": "/customer/burlington-northern-santa-fe",
4     "name": "Burlington Northern Santa Fe",
5     "service_contract_start_date": "03/24/2010",
6     "service_contract_expire_date": "09/24/2015",
7     "hqLatLng": null
8   }
]
  
```

Once your client application has authenticated with the UAA service, you can begin using the Asset service APIs to add and update your asset model data.

This is an example of a POST request that adds an asset to an asset model.

The URL should contain your asset-service-api url

In the example shown here, “locomotive” is a domain object defined in our asset model. The REST request will add this JSON as a locomotive resource to the asset model.

The POST request requires three headers:

1. x-tenant header with your tenant ID
2. Content-Type header that specifies application/json
3. Authorization header that contains a UAA token that was retrieved by a POST request to a UAA service

The body of the request must contain a well-formed JSON to define domain-specific custom asset model objects, add properties to them, define relationships, etc.



Data Management

Lab 5: Using the Asset Service

</Lab>

- Exercise 4: Add Data to the Asset Service
- Exercise 5: Update Domain Objects



© 2016 General Electric Company - All rights reserved.

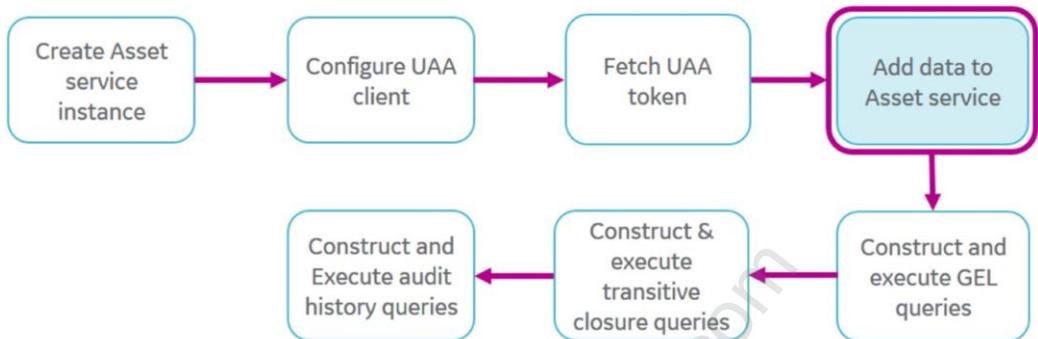
170

Exercise 4: In this exercise, you use the Asset Service APIs to add locomotive asset model data. Our asset model that includes five REST collections: locomotive, fleet, engine, manufacturer, customer and aircraft.

Exercise 5: In this exercise, you will update your new locomotive asset object with a different manufacturer object.



Asset Service Instance Lab Process Flow



© 2016 General Electric Company - All rights reserved.

171



Data Management

Querying Asset Data



© 2016 General Electric Company - All rights reserved.

172

Once asset entities have been added to the asset model, you can query asset entities and their relationships with other asset entities by submitting HTTP requests.



Graph Based Data Model

Graph database stores:

- Asset data
- Syntax structure is a set of **triples** – subject, predicate and object



- Query assets using **Graph Expression Language (GEL)**



© 2016 General Electric Company - All rights reserved.

173

Benefits of a Graph Based Model:

- Ideal for decentralized systems
- Easy to infer context (1st degree neighbors)
- Scalable: Every edge is independently added/removed
- Variety: Simple enough to support a variety of models

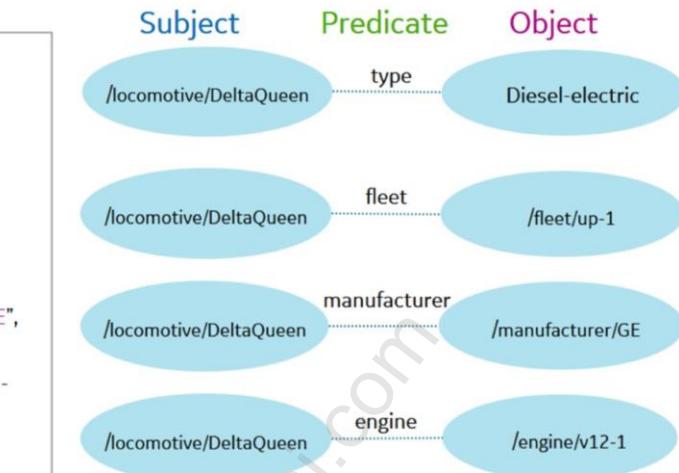
GEL is an acronym for **GraphDB Expression Language**, and is used to query Predix Asset.



Triples - Example

JSON Asset Instance Body

```
[  
  {  
    "uri": "/locomotive/DeltaQueen",  
    "type": "Diesel-electric",  
    "model": "ES44DQ",  
    "serial_no": "00123",  
    "emission_tier": "1+",  
    "fleet": "/fleet/up-1",  
    "manufacturer": "/manufacturer/GE",  
    "engine": "/engine/v12-1",  
    "hpLatLng": {"lat": 33.914605, "lng": -  
117.253374}  
  }  
]
```



© 2016 General Electric Company - All rights reserved.

174

This JSON will generate 3 triplets in the graph store:

In the first triplet, the **subject** is locomotive/DeltaQueen (the uri value), the **predicate** is "fleet" (the attribute **name**) and the **object** is "/fleet/up-1" (the attribute **value**)

The other two triplets that are created are:

/locomotive/DeltaQueen (subject) with a predicate "manufacturer" and an object "manufacturer/GE"

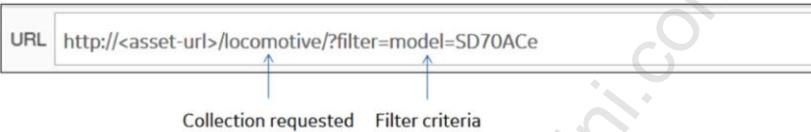
/locomotive/DeltaQueen (subject) with a predicate "engine" and an object "engine/v12-1"



Querying Asset Data

Construct queries against asset data

- Filters data from a result set
 - Implemented with a **filter clause**
- Filter defined for an asset property
 - Append filter clause to asset application URL
 - `http://<asset-url>/domainName/?filter=property operator value`



© 2016 General Electric Company - All rights reserved.

175

A GET request returns all asset objects specified in the URL from the database (or a single asset if an asset Id is specified). You can further refine the set of assets returned by appending a filter clause to the asset application URL in a GET operation.

The syntax for constructing a filter clause using GEL is: `?filter=<property><operator> =<value>` where `<property>` is an attribute defined for the asset.

The filter allows you to filter the data that appears in results. In this example, we are filtering a set of data to return only those locomotive objects with a model of SD70ACe.



GEL Query Supported Operators

Operator	Expression Description	Examples
()	PARENTHESES	/engines?filter=(name=General Electric)<manufacturer
=	EQUALS	/planes?filter=name=747-8
:	AND	/planes?filter=description=Wide-body jet airliner:manufacturer=/manufacturer/Airbus
	OR	/manufacturer?filter=name=Boeing name=ilyushin
..	RANGE	/engines?filter=pressure=100..125
\.	Dot is included in Key value – Must be escaped inquiry Ex: model.id: "76865"	filter=model\\.id=76865
.	Query for data in a nested attribute Ex: "hqLatLng": { "lat": 47.655492, "lng": -117.427025}	filter=hqLatLng.lat=47.655492



By default, the order of operations in a filter clause is from left to right. You can use parenthesis () to override this default and determine the order of operations.

The following operators are supported by GEL:

() - By default, the order of operations is from left to right. Use parentheses to override this. Evaluation is from left to right within the parentheses first. Then the single result is used as a single term for evaluation outside the parentheses.

: (AND) A colon is used because '&' has special meaning in URLs

= (Equals)

| (OR)

... (RANGE between 2 values)

Notice that multiple operators may be combined within a single clause (e.g.

/planes?filter=name=747:manufacturer=Boeing) In this case "=" as well as ":" (and)

The same operator many be used multiple times within a filter clause.

If your JSON Key value includes a dot character, you need to escape the 'dot' in a query to pass the information after the 'dot' in the query to the server. Otherwise, your query will be truncated at the 'dot'

Example:

uri: "/assets/1",

model.id: "76865"

To query this information: filter=model\\.id=76865

If there are nested attributes like in our locomotive data for latitude and longitude, you need to use an un-escaped 'dot' character in your query

Example:

"hqLatLng": { "lat": 47.655492, "lng": -117.427025}

To query the latitude: filter=hqLatLng.lat=47.655492



arun.moses@capgemini.com



Selective Fields Queries

Retrieves selected fields from an object

Use **fields clause** in request method

Syntax: ?fields=field1,field2,...fieldn

http://<asset_instance_uri>/engine?fields=uri,model

Returns only those selected fields for collection

```
GET https://predix-asset.run.aws-usw02-pr.ice.predix.io/locomotiveengine?fields=uri,model
1+
2+
3+ {
4+   "uri": "/locomotiveengine/LOCOMOTIVE_1",
5+   "model": "ES44AC"
6+ },
7+ {
8+   "uri": "/locomotiveengine/LOCOMOTIVE_2",
9+   "model": "SD70ACe"
10+ },
11+ {
12+   "uri": "/locomotiveengine/LOCOMOTIVE_3",
13+   "model": "ES44AC"
14+ },
15+ {
16+   "uri": "/locomotiveengine/LOCOMOTIVE_4",
17+   "model": "AD49SFe"
18+ },
19+ {
20+   "uri": "/locomotiveengine/LOCOMOTIVE_5",
21+   "model": "ES44AC"
22+ }
```



© 2016 General Electric Company - All rights reserved.

177

By default, a GET request will return *all* fields (or attributes) defined for the asset object or objects specified (for example: http://<asset_api_url>/engine)

The **?fields** clause in a request method allows you to retrieve selective fields of a large object or a collection of large objects. For example, if you have an asset with many attributes, but you only want a few attributes returned, you would indicate your selective fields in the field clause.

Fields here refer to attributes defined for an asset.

The syntax for the fields clause is: **?fields=field1,field2,...,fieldn** (fields are separated by commas)

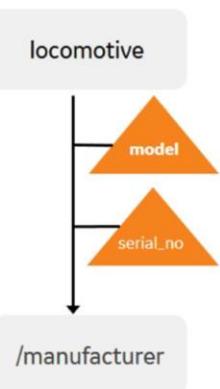
In the example shown here, the fields clause will return a collection of engine assets, but only the uri, type and horsepower fields (attributes) are displayed



GEL Sample Queries

Examples

- Query all locomotives with model of ES44AC or C40-8W
`/locomotive/?filter=model=ES44AC|model=C40-8W`
- Query all locomotives with serial number between 10 and 25
`/locomotive/?filter=serial_no=010..025`
- Query all locomotives with model of ES44AC and manufacturer of GE
`/locomotive/?filter=model=ES44AC&manufacturer=/manufacturer/GE`
- Query parameters can be combined
`/locomotive/?filter=type=Diesel-electric&model=SD70ACe&fields=uri,model`



© 2016 General Electric Company - All rights reserved.

178

Here are some examples of simple GEL queries using the "EQUAL", "AND", and "OR" operators. The first two query examples use properties defined for the locomotive domain object, while the third query uses a property that refers to a *different* domain object (""/manufacturer/GE").

Note the last query example that combines the ?filter and ?fields clauses – you need to add an "&" symbol before the fields clause, but no "?" before the fields clause - a question mark separates the endpoint from the request parameters.



Traversing Relationships

Fleet Relationship

```

{
  "uri": "/locomotive/1",
  "type": "Diesel-electric",
  "model": "ES44AC",
  "serial_no": "001",
  "emission_tier": "0+",
  "fleet": "/fleet/up-1",
  "manufacturer": "/manufacturer/GE",
  "engine": "/engine/v12-1",
  "hqLatLng": {"lat": 33.914605, "lng": -117.253374}
}

{
  "uri": "/locomotive/2",
  "type": "Diesel-electric",
  "model": "SD70ACe",
  "serial_no": "002",
  "emission_tier": "0+",
  "fleet": "/fleet/up-1",
  "manufacturer": "/manufacturer/electric-motive",
  "engine": "/engine/v16-2-1",
  "hqLatLng": {"lat": 47.665492, "lng": -117.427074}
}

```

© 2016 General Electric Company - All rights reserved.

179

Not only can you query assets based on their attributes, but you can query assets based on their relationships. Understanding how to traverse relationships in an asset model will help you when constructing these types of queries.

Based on the way you traverse these relationships, it's either a forward-relate traversal (in the direction of the arrow), or backward-relate traversal (in the opposite direction of the arrow). In this example, locomotive-to-manufacturer would be a **forward-relate** traversal. This is seen in the corresponding JSON where the uri defines the starting point of the traverse and the attribute name, let's select manufacturer, defines the endpoint of the traverse.

Navigating from manufacturer-to-locomotive would be a **backward-relate** traversal.

Multiple assets can be linked to the same object. For example, both locomotive 1 and locomotive 2 assets have a forward-relate fleet relationship to the up-1 fleet object.

As we add more and more asset objects, the graph grows – more relationships and objects get added.



GEL Relate Queries

Queries based on relationships between assets

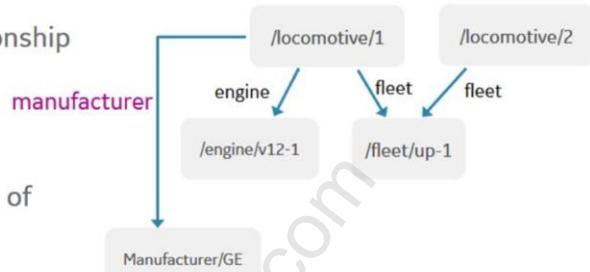
Two ways to traverse relationships

- **Forward-relate**

- Traverse in direction of relationship
- Uses “>” operator

- **Backward-relate**

- Traverse in opposite direction of relationship
- Uses “<” operator



© 2016 General Electric Company - All rights reserved.

180

Forward and backward queries allow you to query relationships of objects stored in your asset model. For example, “which manufacturers make diesel-electric locomotives?”, or “which fleets have ES44AC locomotives?”

Relate queries use the “>” and “<” operators to indicate which direction of the arrow (that represents the relationship between two domain objects) to move

Read the greater than character “>” as **forward-relate**. It selects values stored in the currently selected objects based on predicate (predicate is key). In other words, the new selection is the set of values where object.predicate == value is in the current selection.

The forward relate query uses a property of the Subject domain object of the triple – in this example, for **/locomotive/1** (the **Subject**) there will be a property (attribute) called **manufacturer**.

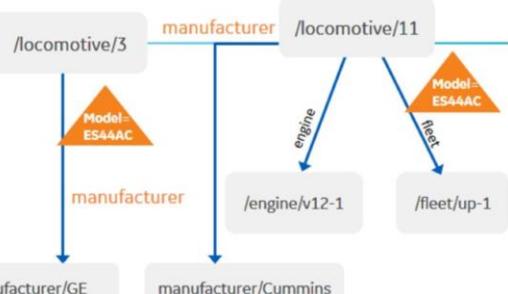
Read the less than character “<” as backward-relate. It selects objects based on values stored in the current selection and predicate in any object. In other words, the new selection is the set of objects where object.predicate = value in current selection.



Forward-Relate Query Logic

Find all the manufacturers for the asset with model =ES44AC

`/manufacturer?filter=model=ES44AC
>manufacturer`



Logic	Syntax	Object(s) Returned
1: Select <u>all</u> objects with model = ES44AC	?filter=model= ES44AC	Locomotive3 Locomotive11
2: Traverse in forward direction over manufacturer relationship	>manufacturer	
3: Return only manufacturer objects	/manufacturer	manufacturer/Cummins/ Manufacturer/GE



© 2016 General Electric Company - All rights reserved.

181

Let's take a look at the logic behind the forward-relate query by examining a sample query that uses a forward-relate operator.

When interpreting queries, always begin at the `?filter=` clause

1) When this query is executed it first looks through entire asset model and finds all objects with Models "ES44AC" (`?filter=model=ES44AC`)

Note: in our sample, this returns only **locomotive** objects, but in the real world, you might have many different types of objects returned (engine, locomotive, fleet, etc.)

2) From the resulting objects selected, traverse in the forward direction over the manufacturer relationship (this is defined by the `>manufacturer` operator) *** in our asset model, *only* locomotive objects have a "manufacturer" relationship, but it's possible that other types of objects have manufacturer relationships ***

3) Out of this new result set, only select manufacturer objects (this is defined by the `/manufacturer` syntax).

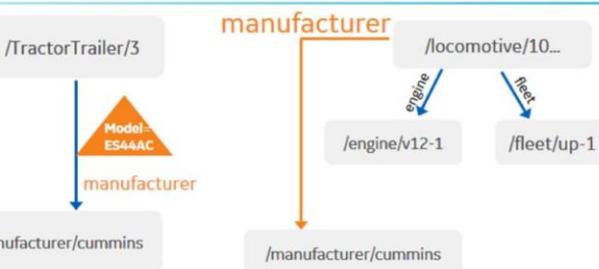
The query returns manufacturers that make ES44AC locomotives which in this example is a General Electric manufacturer asset object and a Cummins object manufacturer.



Backward-Relate Query Logic

Find all the locomotives whose manufacturers have the name Cummins

`/locomotive?filter=name=Cummins <manufacturer`



Logic	Syntax	Object(s) Returned
1: Select <u>all</u> objects with name = Cummins	?filter=name= Cummins	manufacturer/Cummins
2: Traverse in backward direction over manufacturer relationship	<manufacturer	Locomotive/10 and TractorTrailer/3
3: Return only locomotive objects	/locomotive	locomotive/10



© 2016 General Electric Company - All rights reserved.

182

In this backward-relate query example, we are looking for all locomotive assets built by the Cummins manufacturer.

When interpreting queries, always begin at the `?filter=` clause

- 1) When this query is executed it first looks through entire asset model and finds all objects with name of Cummins (`?filter=name=Cummins`)

Note: in our sample, this returns only `manufacturer` objects

- 2) From the resulting manufacturer objects selected, traverse in the backward direction over the manufacturer relationship (this is defined by the `<manufacturer` operator)

- 3) Traversing backwards over the relationship brings you to a locomotive object.
So, only locomotive asset objects are selected. This includes locomotive 10 and possibly others.



Data Management

Lab 5: Using the Asset Service

</Lab>

- Exercise 6: Construct and Execute GEL Queries



© 2016 General Electric Company - All rights reserved.

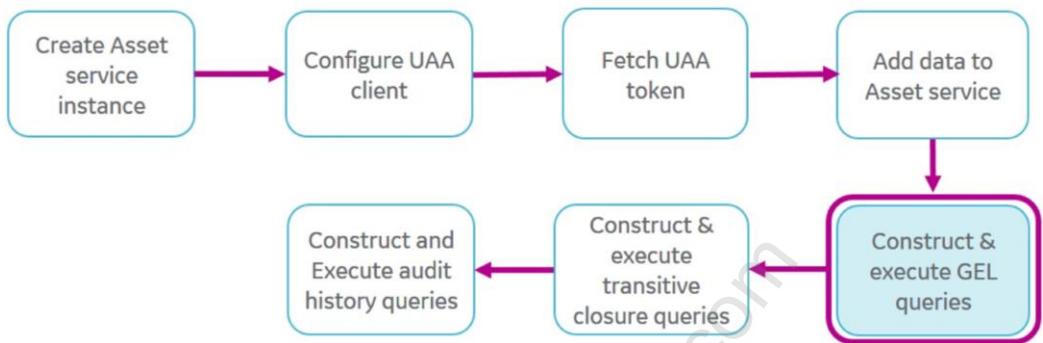
183

Construct and Execute GEL Queries and Transitive Closure Queries

Exercise 6: In this exercise, you construct and execute different GEL queries to control asset data returned by GET requests.



Asset Service Instance Lab Process Flow



© 2016 General Electric Company - All rights reserved.

184

GEL Query Supported Operators – Hierarchical Models

Operator	Expression	Examples
>	FORWARD-RELATE	/planes?filter=name=ilyushin>models
<	BACKWARD-RELATE	/planes?filter=name=Boeing<manufacturer name=A310



© 2016 General Electric Company - All rights reserved.

185

(FORWARD RELATE) If a domain object has a property which is an URI, then use ">" to select what that URI identifies.

< (BACKWARD RELATE) It works just like ">", except that it follows URI properties backward.

>[token] FORWARD-RELATE with transitive closure. Is written as: >[t2] where "2" is the number of levels to traverse UP through the hierarchy model

<[token] BACKWARD-RELATE with transitive closer. Is written as: >[t2] where "2" is the number of levels to traverse DOWN through the hierarchy model



Transitive Closure Queries

Query asset relationships

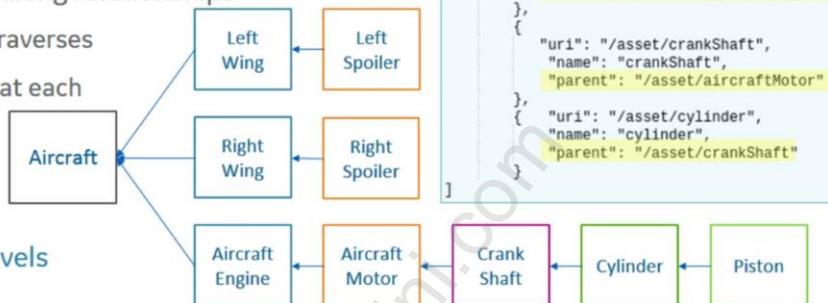
Useful in hierarchical models

- With deep parent-child levels

Traverse multiple levels along relationships

- Forward and backward traverses
- Return **all** objects found at each level

Maximum depth of 10 levels



```
[
  {
    "uri": "/asset/aircraft",
    "name": "GE-Aircraft",
    "parent": null
  },
  {
    "uri": "/asset/aircraftEngine",
    "name": "aircraftEngine",
    "parent": "/asset/aircraft"
  },
  {
    "uri": "/asset/aircraftMotor",
    "name": "aircraftMotor",
    "parent": "/asset/aircraftEngine"
  },
  {
    "uri": "/asset/crankShaft",
    "name": "crankShaft",
    "parent": "/asset/aircraftMotor"
  },
  {
    "uri": "/asset/cylinder",
    "name": "cylinder",
    "parent": "/asset/crankShaft"
  }
]
```

© 2016 General Electric Company - All rights reserved.

186

In this aircraft asset model example, assets are defined in a more hierarchical structure. Reason why we are using “aircrafts” as examples instead of “locomotive” as in previous queries examples.

This model contains assets 6 levels deep. We have the JSON representation that shows how aircraft assets are linked to each other based on the parent relationship. The diagram illustrates the hierarchy – the “aircraft” asset is the parent asset at the top of the structure. Notice in the JSON, the aircraft definition has no parent attribute, which sets it at the top level of the hierarchy. Directly linked to parent aircraft, are the leftwing, rightwing and aircraft engine assets. This relationship is defined in the “parent” attribute of these assets as seen in the JSON. In this asset model example, the hierarchy extends down 5 levels to the piston child asset.

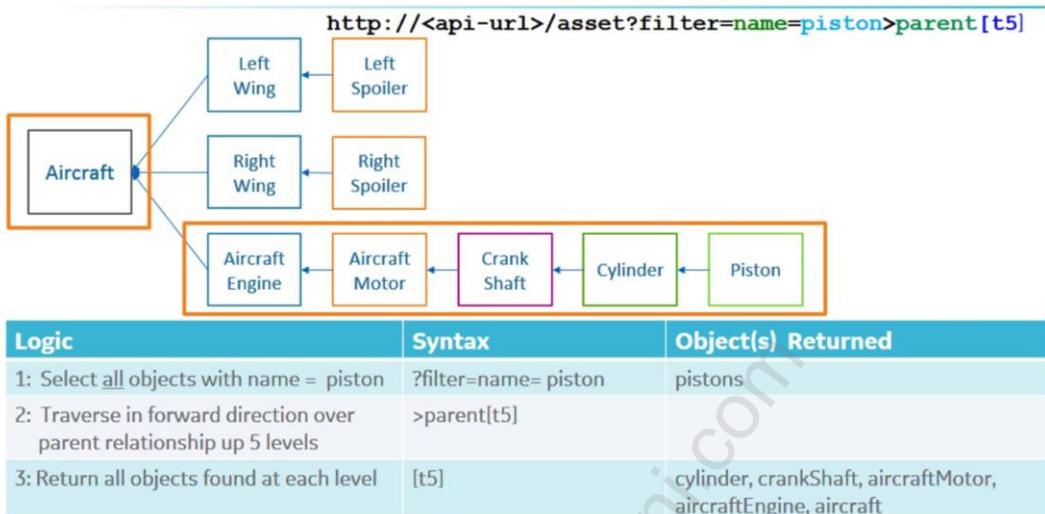
We can traverse the hierarchy forwards and backwards and return all related assets by using GEL transitive closure queries. These queries are designed for hierarchical models with depth –those models that have multiple parent-child relationship levels.

Unlike forward and backward-relate queries that return assets for just one level, transitive closure queries return assets for multiple levels in a hierarchy.

Note: 10 is the maximum depth for a transitive closure query. Specifying a greater number will return an error.



Forward-Relate Transitive Closure Query



© 2016 General Electric Company - All rights reserved.

187

This example examines the execution logic of a forward-relate query with transitive closure. The query shown here retrieves all parent assets of piston asset, moving up five levels in the asset hierarchy. The levels to traverse are defined in the token parameter enclosed in square brackets. The "t" is followed by the number of levels to traverse. The direction to move in is specified with the ">" operator. "asset" is simply the name of a collection that contains all the asset objects shown in the diagram.

An example of a **backward-relate transitive closure** query would be: retrieve all child assets of the aircraft asset moving down 5 levels. The GEL query to use would be: `http://<api-url>/asset?filter=name=GE-Aircraft<parent[t5]`

This returns the following assets: leftwing, leftspoiler, rightwing, rightspoiler, aircraftengine, aircraftMotor, crankShaft, cylinder, piston



Data Management

Lab 5: Using the Asset Service

</Lab>

- Exercise 7: Construct and Execute Transitive Closure Queries



© 2016 General Electric Company - All rights reserved.

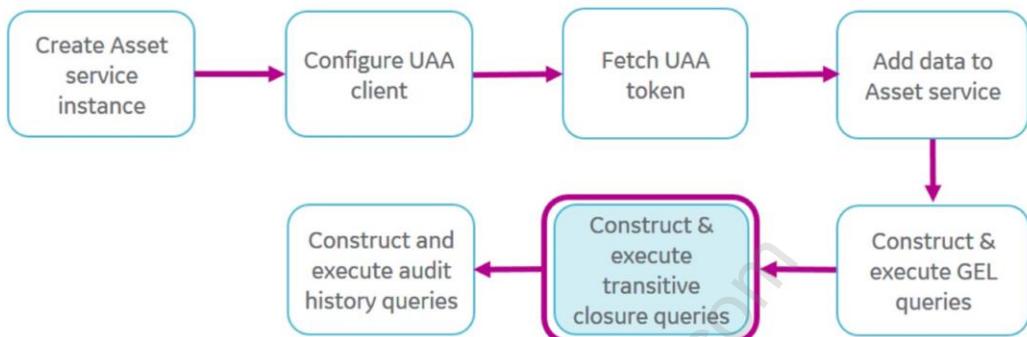
188

Construct and Execute GEL Queries and Transitive Closure Queries

Exercise 7: In this exercise, you work with the asset data model which is more representative of a hierarchical structure containing many levels of objects. You use the transitive closure operator to traverse the asset model and return *all* asset objects for each level specified in the query.



Asset Service Instance Lab Process Flow



© 2016 General Electric Company - All rights reserved.

189



Data Management

Audit History Trail Queries



© 2016 General Electric Company - All rights reserved.

190



Asset Audit History Trail

Tracks **changes** to asset data

- Changes to individual attributes: old value | new value
- Actions taken on objects: UPDATE, CREATE, DELETE

Predefined set of audit attributes

- System-defined attributes
- User-defined attributes

Retrieve audit history with GET requests

Audit history enabled with POST request

- Disabled by default

```
{
  "id": 219405,
  "versionId": -1,
  "timestamp": "2016-07-02T00:08:08.966Z[Etc/UTC]",
  "action": "UPDATE",
  "identifier": "/locomotiveengine/LOCOMOTIVE_99",
  "elementAction": "modified",
  "elementUri": "/locomotiveengine/LOCOMOTIVE_99.emission_tier",
  "elementOldValue": "1",
  "elementNewValue": "2"
},
{
  "id": 219404,
  "versionId": -1,
  "timestamp": "2016-07-02T00:08:08.966Z[Etc/UTC]",
  "action": "UPDATE",
  "identifier": "/locomotiveengine/LOCOMOTIVE_99",
  "elementAction": "modified",
  "elementUri": "/locomotiveengine/LOCOMOTIVE_99.type",
  "elementOldValue": "Electric_Only",
  "elementNewValue": "Electric_Only_HOV"
},
```



© 2016 General Electric Company - All rights reserved.

191

The audit history feature allows you to retrieve changes made to assets store in your Asset repositories. Changes are tracked for individual asset attributes and for the asset object as a whole (e.g. operations performed such as create, update, delete).

By default, the asset audit history tracks a set of system-defined metadata for each asset that includes date and timestamp of the change, actions performed, and old and new values.

Optionally, you can use a set of user-defined attributes to track information that is specific to your business. For example, tracking the userid of the individual who made the change, or the reason for the asset update ("swapped out expired parts").

Audit history objects are stored in a separate database from asset data. You query audit objects using GEL queries – just as you do with asset objects in your data model.

The Audit History functionality is disabled by default. Users must make a REST API call to the endpoint "/system/configs" to enable and disable the service explicitly.

Best practice is to ensure that the Audit History function is disabled before a large upload. Once the upload is complete you are able to re-enable the Audit History Function.



Audit History Trail Query Examples

- History records for collection or individual assets
- History records within a date range
- Records for an asset on a specific date

Audit Parameters	Examples	Description
/audit	https://<asset_uri>system/audit	All audit records
userId	https://<asset_uri>system/audit?filter=userId=<userId>	Who updated the record
Before/after	https://<asset_uri>system/audit?filter=before<UTC>;after=<UTC>	All records in a specific time period
Action	https://<asset_uri>system/service/audit?filter=action=DELETE or action=UPDATE	Items that have deleted or updated
Reason	https://<asset_uri>system/service/audit?filter=reason=<some reason>	List of records matching the given reason



© 2016 General Electric Company - All rights reserved.

192

Here are some additional examples of queries you can run using Audit History Trail



JSON Schema

Validate your domain objects against a pre-defined schema

Schemas can be created for any asset

Requirements:

- Must be Valid JSON
- Must have type = 'Object'
- Must have a URI property

For more information visit www.json-schema.org

```

1 [ 
2 {
3   "title": "Employee Data",
4   "type": "object",
5   "properties": {
6     "uri": {"type": "string"},
7     "id": {
8       "description": "employee ID",
9       "type": "integer",
10      "minimum": 0
11    },
12    "firstName": {
13      "type": "string"
14    },
15    "lastName": {
16      "type": "string"
17    },
18    "age": {
19      "description": "Age in years",
20      "type": "integer",
21      "minimum": 0
22    },
23    "email": {
24      "description": "best email contact",
25      "type": "string"
26    }
27  },
28  "required": ["uri", "firstName", "lastName", "email"]
29 }
30 ]

```



© 2016 General Electric Company - All rights reserved.

193

JSON Schema follows the standard JSON schema architecture. For more information on this, please visit json-schema.org

JSON Schema allows you to validate your domain objects against a pre-defined schema. Schemas can be created for each domain object.

In this example, we can see the requirements for 'type' and uri are included. Under properties, in addition to the required uri field, we have included attribute fields and their expected formatting.

Once you have completed listing the attributes and their formatting standards, you can list the required attributes for each asset/object instance



Predix Asset Scripting Engine

- Bind custom business logic to the Predix Asset RESTful API
- Allows for pre and post processing
- Allows non-technical business users to define business rules and interact with Asset repositories
- Business Logic task examples
 - Validate and accept/reject requests
 - Modify contents of incoming requests
 - Perform operations after a request has been processed
 - Execute Ad hoc Processing and return a custom response



© 2016 General Electric Company - All rights reserved.

194

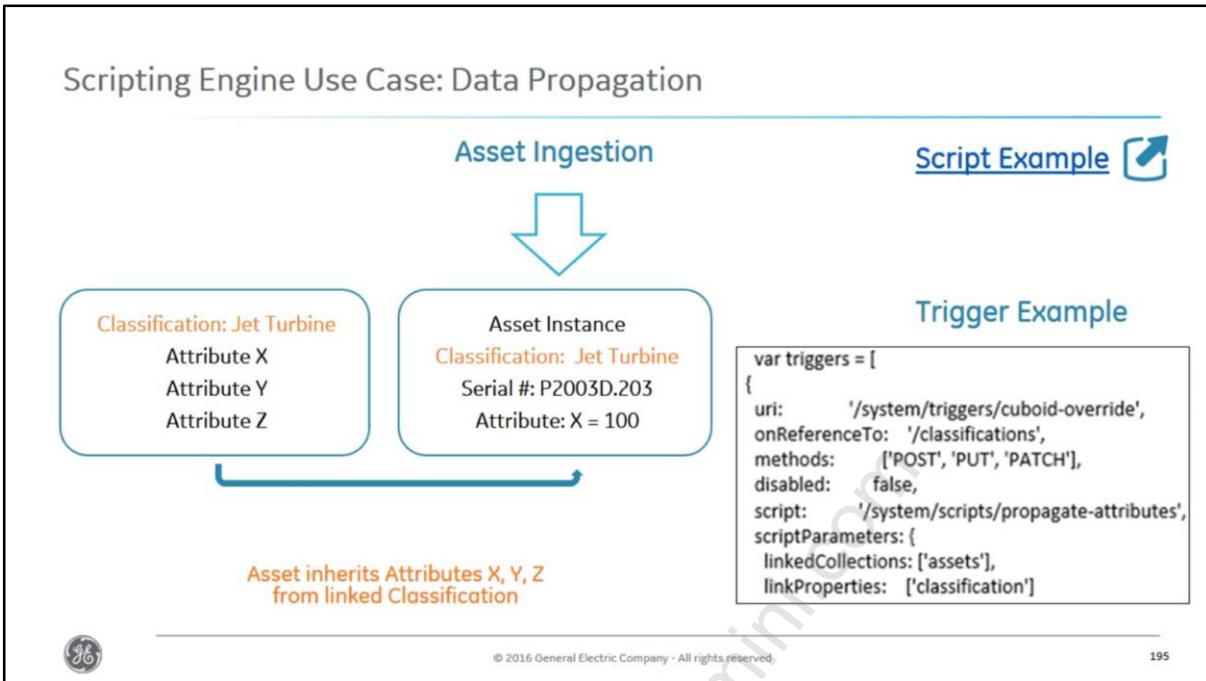
The Predix Asset Scripting Engine is a service that allows users to bind custom business logic to the Predix RESTful API. The logic can be processed before or after a request has been processed by Predix Asset.

Allowing business logic to be added to asset queries enables non-technical business users to define rules and interact with asset repositories

Some specific examples of business logic include:

- Rejecting requests that do not match a schema
- Prevent specific resources from being created, read, updated, or deleted
- Create, Update or delete resources after processing a request
- Read from multiple Predix Asset resources and generate a report





[Scripts](#) and [Triggers](#) are the building blocks that allow clients to define and apply their custom business logic. Scripts contain the client's business logic and Triggers determine when that business logic is executed. Both Scripts and Triggers are REST resources that can be created, read, updated, and deleted by clients. Clients first create a set of Scripts containing their business logic and then create one or more Triggers to bind that business logic to either the Predix Asset REST interface or a custom endpoint.

Here we are posting asset data for an asset instance with a classification attribute of Jet Turbine. If we have already defined the attributes of Jet Turbine, we are able to bind a script and trigger to the post method that will have our new asset instance inherit the predefined attributes from the classification '[Jet Turbine](#)'

In the script example:

In the trigger example, we see the URI location of the trigger, the methods that will trigger the script: 'POST, PUT, PATCH'.

We see the script location and title, and which collections and properties will be updated.



Data Management

Lab 5: Using the Asset Service

</Lab>

Exercise 8: Query Asset Audit History



© 2016 General Electric Company - All rights reserved.

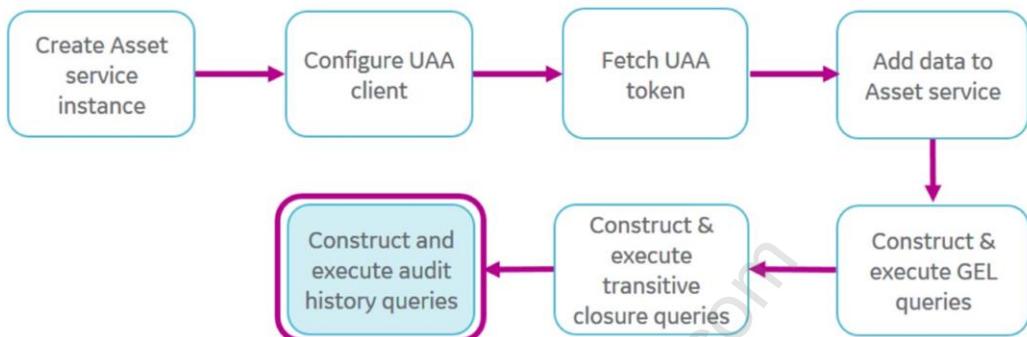
196

Construct and Execute Audit History Queries

Exercise 8: In this exercise, you use the Audit History feature to view changes made to your asset model.



Asset Service Instance Lab Process Flow



© 2016 General Electric Company - All rights reserved.

197



Data Management

PostgreSQL



198



© 2016 General Electric Company - All rights reserved.



GE Digital Predix Training

Module: Data Management

198

Relational Database (PostgreSQL)

- PostgreSQL as a service object-relational database management system (RDBMS)
- Store data securely for retrieval at the request of other software apps
- Handle workloads ranging from single-machine apps to internet-facing apps with many concurrent users
- The data stores plus the ingestion pipeline needed to bring data in for analysis

SQL Database

Store data securely, support best practices, and facilitate retrieval at the request of other software apps.

PostgreSQL



© 2016 General Electric Company - All rights reserved.

199

- Although there is a proliferation of so-called NoSQL databases for special purposes like time series data management, SQL databases are still the most commonly used.
- PostgreSQL is an open source relational database management system (RDBMS)
- The data stores, plus the ingestion pipeline, are the data services that are needed by developers to bring data in for analysis



Locating Postgres Environment Variables

```
File Edit View Search Terminal Help
"postgres": [
  {
    "credentials": {
      "ID": 0,
      "binding_id": "6baab3ee-7c65-4490-883d-a0adf62290ce",
      "database": "de1c46ce8db0e4924acaf115d40ab3dfa",
      "dsn": "host=10.72.6.143 port=5432 user=u3c68d8e8d58f4a02b58126a8c19c8c83
password=9687b12515c54107892150a7670b416e dbname=de1c46ce8db0e4924acaf115d40ab3dfa
connect_timeout=5 sslmode=disable",
      "host": "10.72.6.143",
      "instance_id": "28af8fc5-1ae3-4329-9ec4-495fe372648d",
      "jdbc_uri": "jdbc:postgresql://10.72.6.143:5432/de1c46ce8db0e4924acaf115d40ab3dfa?user=u3c68d8e8d58f4a02b581
26a8c19c8c83\u0026password=9687b12515c54107892150a7670b416e\u0026sslBuild and install=Build
and install=false",
      "password": "9687b12515c54107892150a7670b416e", ...
    }
  }
]
```



© 2016 General Electric Company - All rights reserved.

200

Predix Cloud PostgreSQL uses VCAP_SERVICES environment variables to communicate with a deployed application about its environment.



Data Management

Key Value Store – Redis Service



© 2016 General Electric Company - All rights reserved.

201



Redis - Key-Value Store

- redis is a fast in-memory key value datastore
- Non-relational, distributed, open-source and horizontally scalable
- redis supports most datatypes, data replication and save data on disk
- Used in a number of use cases like caching, messaging-queues

Key-Value Store

Use this open-source key-value cache and store that can contain strings, hashes, lists. Sets, bitmaps, and more.



© 2016 General Electric Company - All rights reserved.

202

Redis natively supports Publish/Subscribe, as well as Short lived data in your app like web application sessions or web page hit counts

Features:

Next Generation Databases: Redis is an in-memory key value datastore written in ANSI C programming language: Non-relational, distributed, open source and horizontally scalable.

Exceptionally Fast : Redis is very fast. It's used quite often as a cache that can be rebuilt from a backing alternative primary database for this reason.

Supports Rich data types : Redis natively supports most of the datatypes that most developers already know like list, set, sorted set, hashes. This makes it very easy to solve a variety of problems **because we know which problem can be handled better by which data type.**

Operations are atomic : All the Redis operations are atomic, which ensures that if two clients concurrently access Redis server will get the updated value.

MultiUtility Tool : Redis is a multi utility tool and can be used in a number of use cases like caching, messaging-queues (Redis natively supports Publish/Subscribe), any short lived data in your application like web application sessions, web page hit counts, etc.



Redis Data Structures

Value	Data Structure
List	Collection of string elements sorted in order of insertion: linked lists
Sets	Collection of unique, unsorted string elements
Sorted Sets	Similar to Sets - every string element is associated to a floating number value, called score Elements always sorted by their score
Hashes	Maps composed of fields associated with values Both the field and value are strings Similar to Ruby or Python hashes
Bit Arrays	Handle string values like an array of bits
HyperLogLog	Probabilistic data structure used to estimate cardinality of a set



© 2016 General Electric Company - All rights reserved.

203

Redis is not a *plain* key-value store, actually it is a *data structures server*, supporting different kind of values. What this means is that, while in traditional key-value stores you associated string keys to string values, in Redis the value is not limited to a simple string, but can also hold more complex data structures.

Some of its additional features:

- It's a compelling replacement for Memcached allowing more advanced caching for the different kinds of data you store. Like Memcached, everything is held in memory. Redis does persist to disk, but it doesn't synchronously store data to disk as you write it.
- Redis not only supports string datatype but it also supports list, set, sorted sets, hashes datatypes, and provides a rich set of operations to work with these types. If you have worked with Memcached (an in-memory object caching system) you will find that it is very similar, but Redis is Memcached++. Redis not only supports rich datatypes, it also supports data replication and can save data on disk.
- Uses associative array data model where each key is associated with one and only one value in a collection. This relationship is referred to as a key-value pair.



Data Management

RabbitMQ Service



© 2016 General Electric Company - All rights reserved.

204



RabbitMQ

- Persistent, reliable messaging between applications, components, and devices
- Supports clients in many languages:
 - Java, .NET, Ruby, python, etc.
- Exchanges supported:
 - Built-in direct
 - Fanout
 - Topic
 - Header exchange types

Message Queue (AMQP)

Use this scalable, high-performance, multi-Protocol tool to message between apps, components, and devices.



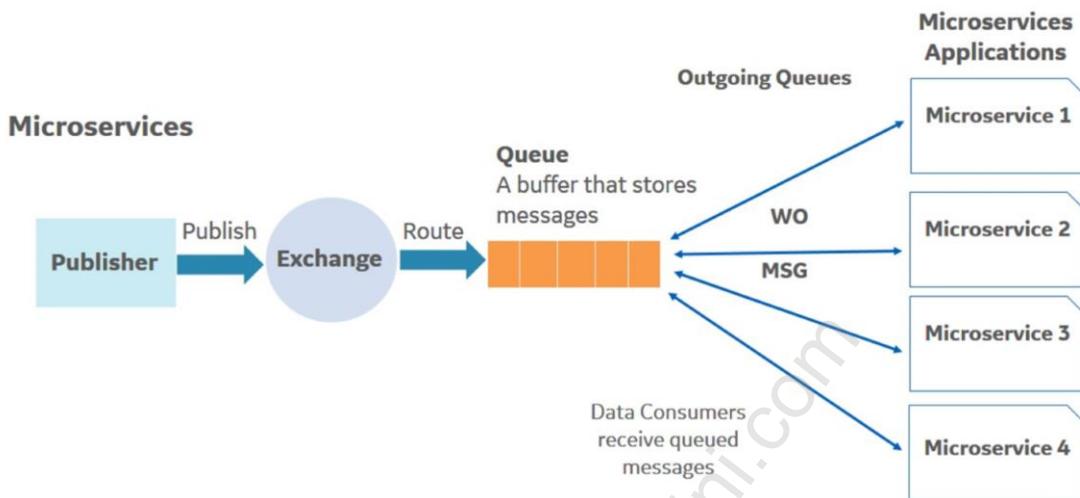
© 2016 General Electric Company - All rights reserved.

205

Use this message broker software service to provide persistent, highly available, reliable messaging between applications, components, and devices. The message queue service supports a variety of messaging protocols, many clients, and flexible routing with built-in direct, fanout, topic, and header exchange types.



Using RabbitMQ



RabbitMQ is a messaging broker - an intermediary for messaging. It gives your applications/microservices a common platform to send and receive messages, and your messages a safe place to live until received.

Messages are published to *exchanges*, which are often compared to post offices or mailboxes. Exchanges then distribute message copies to *queues*. Once a message is stored in the queue it is ready to be sent out to the consumers (applications or mobile devices).

Think of field technicians receiving and sending work orders through RabbitMQ queue messaging system. The broker either deliver messages to consumers subscribed to queues (microservices/applications/SDK Mobile Devices), or consumers fetch/pull messages from queues on demand.

Create a RabbitMQ Service Instance

File Edit View Search Terminal Help

Syntax: cf create-service p-rabbitmq-35 <Plan> <your_key-value_instance>

Syntax: cf bind-service <your_app_name> <your_message_queue_instance>



© 2016 General Electric Company - All rights reserved.

207

Create a RabbitMA service instance through the CLI or through the catalog at predix.io.

Create a Redis service instance with the command:

cf create-service p-rabbitmq-35 <Plan> <your_message_queue_instance>

You must bind your application to your RabbitMQ service instance to provision connection details for your service instance in the VCAP environment variables. Cloud Foundry runtime uses VCAP_SERVICES environment variables to communicate with a deployed application about its environment.

1. Bind your application to the service instance you created.
cf bind-service <application_name> <your_message_queue_instance>
2. Restage your application to ensure the environment variable changes take effect:
cf restage <application_name>
3. To view the environment variables for your application, enter:
cf env <application_name>

The command shows the environment variables, which contain your basic authorization credentials, client ID, and the endpoint URI.



Data Management

Time Series



© 2016 General Electric Company - All rights reserved.

208

The Asset Data service provides you with the tools to create and manage asset models in the cloud. But before we can use those tools, we need to understand the concept of asset modeling.



Time Series

- Enables quick/efficient ingest, store and analyze time series data
- Query data using groups, time ranges, values, aggregations, qualities, filters
- Sequence of data points collected at set time intervals

Features:

- Indexing data for quick retrieval
- Millisecond data point precision

Time Series

Quickly and efficiently manage, ingest, store, and analyze data.

PREDIX



© 2016 General Electric Company - All rights reserved.

209

Time Series data is a sequence of data points collected at set time intervals over a continuous period of time. Sensor data is an example of a common way to generate time series data. A Time Series data store requires a measurement with a corresponding timestamp. The Time Series service provides an attributes field to include additional relevant details about that specific data point, such as units or site, for example, "Site":"SanFrancisco".

The Time Series service provides the following benefits:

- Efficient storage of time series data.
- Indexing the data for quick retrieval.
- High availability so you can access and query your data from anywhere via HTTP.
- Horizontal scalability.
- Millisecond data point precision.



Time Series Service - Features

- **Columnar storage:** a query-efficient format
 - Data Ingestion provides data streaming and support for industrial data formats
- **Data Query** – allows you to query ingested data
 - Supports grouping of data points by tags, time ranges, or values
 - Supports aggregations
 - Filter by attributes to narrow results



© 2016 General Electric Company - All rights reserved.

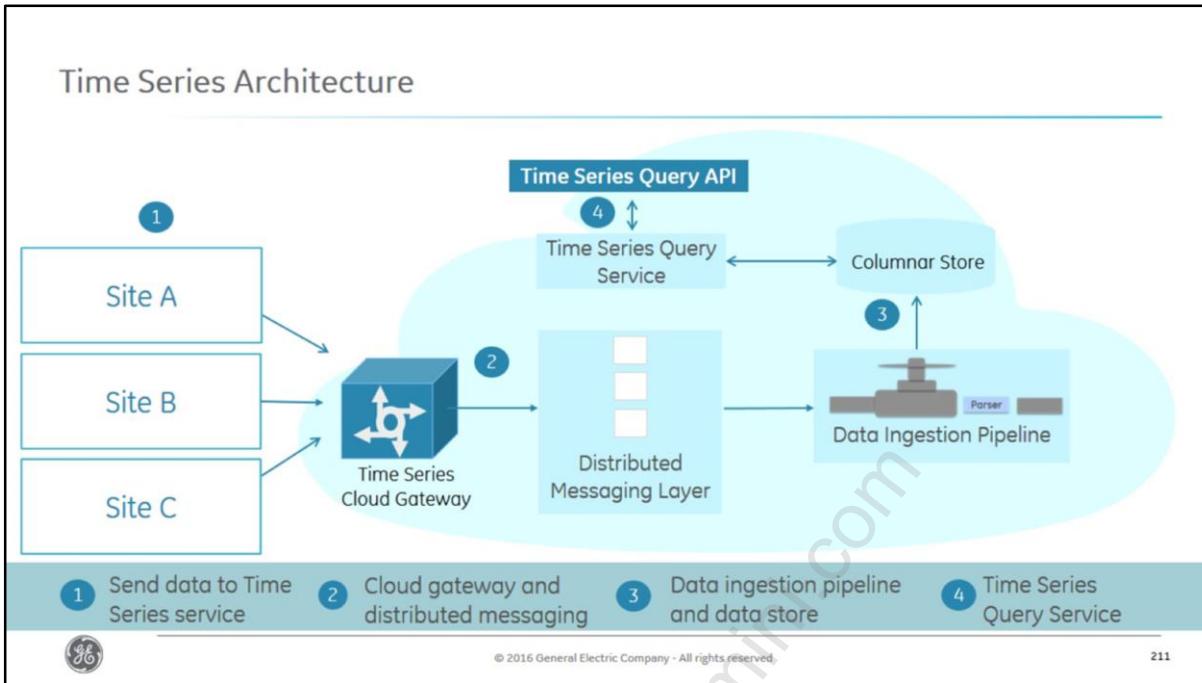
210

Time Series is a single service with two components:

- **Data Ingestion:** The data-ingestion layer provides the ability to ingest streaming data
- **Data Query:** The query API allows you to query data, with support for grouping of data points by tags, time ranges, or values, as well as aggregations. You can also filter by attributes to narrow your results

The VCAP_SERVICES will retrieve 2 different URLs and Zone Ids one for each endpoint (data ingestion and data query)





The high-level architecture diagram shows different ways you can send data to the Time Series service for ingestion, as well as how users can interact with the query service.

The numbered sequence represents the time series data lifecycle (explore in detail in the next slides):

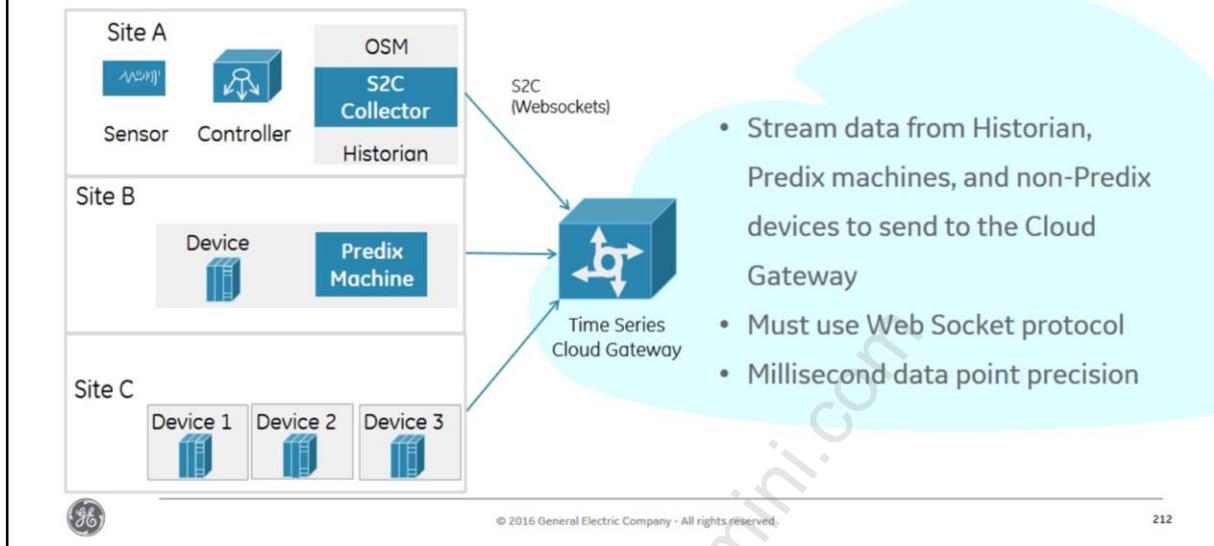
1. Send data to Time series service
2. Cloud Gateway
3. Data Ingestion Pipeline
4. TS Query Service and Data Store

Multiple sites stream live data for ingestion, which is split and sent to the Data Ingestion Pipeline. The data packets are combined and checked to ensure the format is correct. The data is stored in the Columnar Store, available for query through the Time Series Query API.

Time Series is excellent for any purpose where you need to collect a timestamp with a number. For example, temperature, pressure reading, miles-per-hour documented with a timestamp. Time Series is not meant for general relational database use cases, such as an employee database, or document control storage.

Uncompliant data is stored for a limited period of time for diagnostic purposes.

1. Time Series Cloud Gateway



Sites A, B, and C show the different types of machines sending data to the Time Series Cloud Gateway. The WebSocket protocol is used rather than HTTP because it can ingest a higher volume of data, which increases performance.

Site A

Sensors produce time series data associated with a tag name and send it to Historian for storage and management. Server to Client (S2C) subscribes to tags and collects generated data from those tags only. S2C requests a WebSocket connection from a gateway application that is used for data ingestion.

Site B

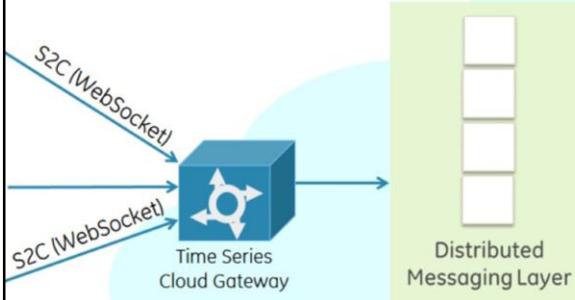
The WebSocket River establishes a connection when you first attempt a data transfer and keeps that channel open for as long as possible. Each data transfer verifies that the WebSocket connection is open. If the connection has been closed, the service opens a new connection.

Site C

Devices use an application to communicate directly with the WebSocket.



2. Distributed Messaging



- In cloud gateway, data is split and distributed through parallel nodes
- Super fast/highly scalable
- Guarantees data processing
- Data structure agnostic



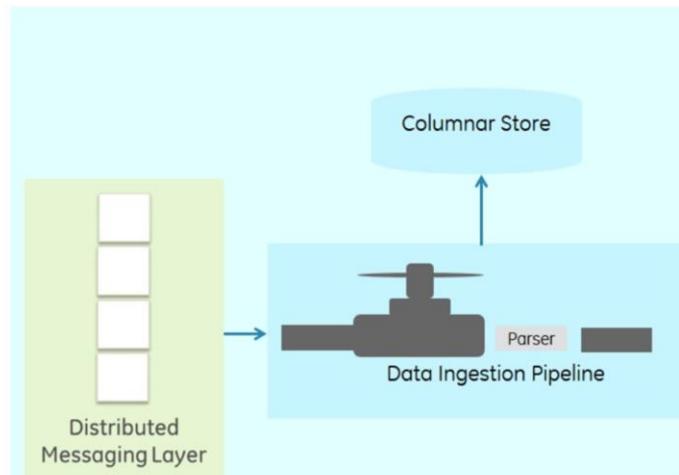
© 2016 General Electric Company - All rights reserved.

213

- Once data reaches the cloud gateway, it is split and distributed through a parallel nodes
- Simple and Scalable as a Cloud Foundry Application
- Optimized Wire Protocol – super fast getting from point to point
- Data structure Agnostic – not married to any particular data structure
- Within the GE network we get 250 micro second performance for a datapoint



3. Data Ingestion Pipeline & Data Store



- Data flows from distributed messaging layer into ingestion pipeline
- Ingestion pipeline puts data back together, checks format is correct and sends it to the store
- Once in the store – data is available for query



© 2016 General Electric Company - All rights reserved.

214

Distributed Messaging Layer

Data moves from the gateway and is split, parallelizing the flow through various nodes to promote efficient performance.

Data Ingestion Pipeline

The pipeline ensures that data is compliant with the ingestion requirements (are the established data parameters in place?) and sends it to the Columnar Store. If it is not compliant, it is stored until fixed.

Columnar Store

Data available for query.



What is an Ingestion Pipeline?

Incoming data needs to be in a certain format

The validation is performed asynchronously in the data ingestion pipeline

- Compliant data moves to columnar store
- Non-compliant data is inaccessible (on roadmap)



© 2016 General Electric Company - All rights reserved.

215

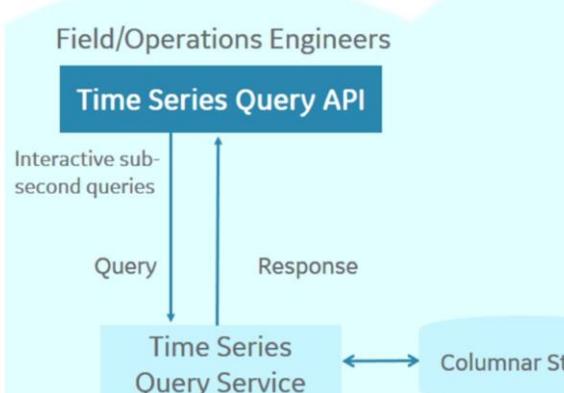
When the user/client sends in the data, we don't immediately validate the format – we simply store it in the distributed messaging layer

Data Ingestion: Where does noncompliant data go?

- The incoming data needs to be in a certain format
- When the user/client sends in the data, we don't immediately validate the format – we simply store it in the distributed messaging layer
- The validation is performed asynchronously in the data ingestion pipeline
- If during that validation we find that the data doesn't comply with the format described, we don't push it into the columnar store but put it in a separate storage
- Currently, users have no way to access that "invalid" data from the APIs. It is in our roadmap to allow users to see that data.



4. Time Series Query API



- RESTful APIs
 - Filter data (i.e. Quality, attributes, measure, time)
 - Support for interpolation
- Data used for predictive analytics



© 2016 General Electric Company - All rights reserved.

216

- Field / Operations Engineers use web portals, dashboards, reports, and charts through computers and mobile devices to query data through the Store
- Analysts use data for predictive analytics

Time Series Query API

Querying is handled through Restful API's.

Time Series Query Service

Parse JSON Body, token checks, queries and returns.



Create a Time Series Service Instance

The screenshot shows a terminal window with a black header bar containing the menu items: File, Edit, View, Search, Terminal, Help. Below the header is a command-line interface. The command is:

```
Syntax: cf create-service predix-timeseries <Plan> <your_TimeSeries_instance>
-c '{"trustedIssuers":["<uaa_issuerId>"]}'
```

Annotations with arrows point to the following parts of the command:

- An arrow points from the placeholder <uaa_issuerId> to the text "IssuerId from VCAP SERVICES".
- An arrow points from the placeholder <your_TimeSeries_instance> to the text "Service instance name".



© 2016 General Electric Company - All rights reserved.

217

Create a time series service instance.

On Mac OS and Linux:

- cf create-service predix-timeseries <plan> <your_time_series_instance> -c '{"trustedIssuers":["<uaa_instance1_issuerId>", "<uaa_instance2_issuerId>"]}'

On Windows:

- cf create-service predix-timeseries <plan> <your_time_series_instance> -c "{\"trustedIssuers\":\\[\"<uaa_instance1_issuerId>\", \"<uaa_instance2_issuerId>\"]}"
- Where:<plan> – the plan associated with a service
- <your_time_series_instance> – the service instance you are creating
- -c: in-line JSON object puts the value of <uaa_issuerId> into the trustedIssuers parameter
- "trustedIssuers" – the issuerID of your trusted issuer (UAA instance)
- For example, <https://13fa0384-9e2a-48e2-9d06-2c95a1f4f5ea.predix-uaa.grc-apps.svc.ice.ge.com/oauth/token>
- You can use a comma-separated list to specify multiple trusted issuers.



Bind App to the Service Instance

File Edit View Search Terminal Help

Syntax: cf bind-service <your_app_name> <your_time_series_instance>



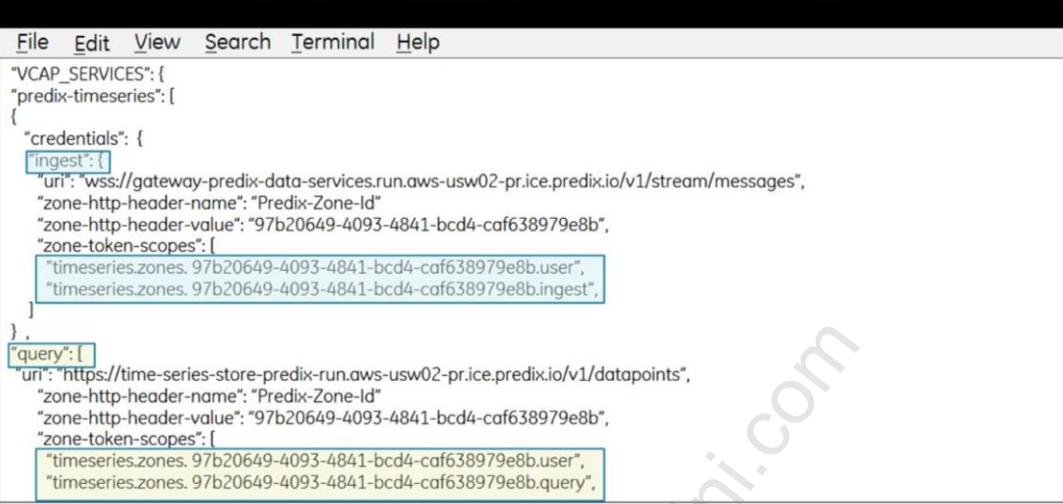
© 2016 General Electric Company - All rights reserved.

218

Bind your application to the Time Series service instance using the bind-service command. You can retrieve this URL from the VCAP_SERVICES environment variable after binding your UAA instance to an application.



Locating Time Series Environment Variables



```

File Edit View Search Terminal Help
"VCAP_SERVICES": {
  "predix-timeseries": [
    {
      "credentials": {
        "ingest": [
          {
            "uri": "wss://gateway-predix-data-services.run.aws-usw02-pr.ice.predix.io/v1/stream/messages",
            "zone-http-header-name": "Predix-Zone-Id",
            "zone-http-header-value": "97b20649-4093-4841-bcd4-caf638979e8b",
            "zone-token-scopes": [
              "timeseries.zones.97b20649-4093-4841-bcd4-caf638979e8b.user",
              "timeseries.zones.97b20649-4093-4841-bcd4-caf638979e8b.ingest"
            ]
          },
          {
            "query": [
              {
                "uri": "https://time-series-store-predix-run.aws-usw02-pr.ice.predix.io/v1/datapoints",
                "zone-http-header-name": "Predix-Zone-Id",
                "zone-http-header-value": "97b20649-4093-4841-bcd4-caf638979e8b",
                "zone-token-scopes": [
                  "timeseries.zones.97b20649-4093-4841-bcd4-caf638979e8b.user",
                  "timeseries.zones.97b20649-4093-4841-bcd4-caf638979e8b.query"
                ]
              }
            ]
          }
        ]
      }
    }
  ]
}

```

© 2016 General Electric Company - All rights reserved.

219

To enable applications to access the Time Series service, your JSON Web Token (JWT) must contain the Predix zone token scopes.

Best practice: creating two clients, one for ingestion and one for query. Then, update with the timeseries.zones user, ingest and query zone-token-scopes, as appropriate.

Ingest – timeseries.zones.xxx.user; timeseries.zones.xxx.ingest
 Query – timeseries.zones.xxx user; timeseries.zones.xxx query



Create Clients with Time Series Authorities

```
File Edit View Search Terminal Help
Client name
Syntax: $ uaac client update <client name>
--authorities "<existing authorities>,
timeseries.zones.<Predix-Zone-Id>.user, timeseries.zones.<Predix-Zone-Id>.ingest,
timeseries.zones.<Predix-Zone-Id>.query"
```



© 2016 General Electric Company - All rights reserved.

220

To enable applications to access the Time Series service, your JSON Web Token (JWT) must contain the Predix zone token scopes:

Data ingestion scopes

timeseries.zones.<Predix-Zone-Id>.user
timeseries.zones.<Predix-Zone-Id>.ingest

Data queries scopes

timeseries.zones.<Predix-Zone-Id>.user
timeseries.zones.<Predix-Zone-Id>.query



Time Series Data Ingestion

A time series dataset uses tags

Tags often represent sensors

A tag consists of:

- Tag Name (Sensor)
- TimeStamp (Time)
- Measure (Value)
- Attributes (key/value pairs)

Tag Name	Measure	Quality
1404144823000	92.85	2
1404144833000	92.95	3
1404144838000	92.82	3
1404144843000	93.12	1
1404144848000	93.48	2
1404144853000	93.48	2
1404144863000	92.93	3

Attributes

Sensor Serial Number
58b6-21C5
Location
ST5



© 2016 General Electric Company - All rights reserved.

221

- A time series dataset uses tags, which are often utilized to represent sensors (for example, a temperature sensor).
- A tag consists of one Tag Name (Sensor), a TimeStamp (Time), a Measure (Value) and, optionally, one or more attributes (key/value pairs).
- The ingestion pipeline enables a developer to take the raw data stream and map it to these data structures before being persisted in the Time Series data store.

Term Definition

Tag name (Required) Name of the tag (for example, "temperature") The tag name can contain only alphanumeric characters, periods (.), slashes (/), dashes (-), and underscores (_).

Measure (Required) Reading (Value) Numeric (for example: 98°C)

Qualities (Optional) Quality of the data, represented by the values 0, 1, 2, 3.

Timestamp (Required) Time Epoch (UNIX epoch time in milliseconds)

Attribute (Optional) Any data associated with a tag (for example: AircraftID=230, Manufacturer = GE). This is useful for filtering data.

Attributes can contain only alphanumeric characters, periods (.), slashes (/), dashes (-), and underscores (_). Attributes cannot contain colons (:) or equal signs (=) as values.

You can use mixed data types, including null and string in data ingestion



Ingestion Tags

Request Headers:

- GET
- POST
- PUT
- PATCH
- DELETE

Raw Form

Add a new header

Authorization	Bearer oJSUihtGt8j5Nh94fth5OVe
Content-type	JSON
Predix.Zone.Id	644a721d-b85a-4094-bc17-f45k...

Request Payload:

```
{
  "messageId": "<MessageID>",
  "body": [
    {
      "name": "<Unique Tag Name>",
      "datapoints": [
        [
          <EpochInMs>,
          <Measure>,
          <Quality>
        ]
      ],
      "attributes": [
        "<AttributeKey>": "<AttributeValue>",
        "<AttributeKey>": "<AttributeValue>"
      ]
    }
  ]
}
```

© 2016 General Electric Company - All rights reserved.

222

This is an ingestion request to REST Client

Data Quality (Optional)

You can tag data with values that represent data quality, which enables you to query by quality in cases where you want to retrieve or process only data of a specific quality. Using data quality can help to ensure that the data you get back in the query result is the most relevant for your purpose.

- | | |
|---|---|
| 0 | Bad Quality |
| 1 | Uncertain Quality |
| 2 | Not applicable |
| 3 | Good Quality If you do not specify quality, the default is good (3). |



Streaming Data Ingestion Using Web Sockets Protocol

- Ingest higher volume of data than HTTP
 - Bidirectional conversation between browser and server keeps connection open
 - Ingestion endpoint format: WSS://<INGESTION_URL>

```
"VCAP_SERVICES": {  
  "predix-timeseries": [  
    {  
      "credentials": {  
        "ingest": {  
          "uri": "wss://gateway-predix-data-services.run.aws-usw02-pr.ice.predix.io/v1/stream/messages",  
          "zone-http-header-name": "Predix-Zone-Id"  
        },  
        "zone-http-header-value": "97b20649-4093-4841-bcd4-caf638979e8b",  
        "zone-token-scopes": [  
          "timeseries.zones. 97b20649-4093-4841-bcd4-caf638979e8b.user",  
          "timeseries.read. 97b20649-4093-4841-bcd4-caf638979e8b.user",  
          "timeseries.write. 97b20649-4093-4841-bcd4-caf638979e8b.user"  
        ]  
      }  
    }  
  ]  
}
```



© 2016 General Electric Company - All rights reserved

223

The time series gateway uses the WebSocket protocol for streaming ingestion. The ingestion endpoint format is: `wss://ingestion url`

The Web Socket protocol is used rather than HTTP because it can ingest a higher volume of data, which increases performance. WebSocket Protocol is an independent TCP-based protocol (Transmission-Control Protocol). The WebSocket protocol makes more interaction between a browser and a website possible, facilitating the real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way a two-way (bi-directional) ongoing conversation can take place between a browser and the server. The communications are done over TCP [port](#) number 80, which is of benefit for those environments which block non-web Internet connections using a [firewall](#).

For all ingestion requests, use the token you received from UAA in the Authorization: section of the HTTP Header in the form of Bearer <token from trusted issuer>.

Note: You can use mixed data types, including null and string in data ingestion.



Time Series Data Consumption

Time Series provides REST APIs for querying and aggregating time series data

Features

- Query time-series data specifying tags (sensors, etc.) and a time window
- Filter, retrieve tags, attribute values and attribute keys from time-series database
- Aggregates, interpolates groups data points in a given time window
- Order queries by timestamp



© 2016 General Electric Company - All rights reserved.

224

The Time Series service provides REST APIs for querying and aggregating time series data.

Features:

- Query time series data specifying tags (sensors, etc.) and a time window (start/end time)
- Filter by attribute values
- Retrieve all tags and attribute keys from the time series database
- Add aggregates and interpolate data points in a given time window
- Group data by attribute values



Time Series API Documentation

API Documentation

▼ Time Series

GET	/v1/aggregations	Get all available aggregations
GET	/v1/datapoints	Query Datapoints
POST	/v1/datapoints	Query Datapoints
GET	/v1/datapoints/latest	Query for Current Value
POST	/v1/datapoints/latest	Query for Current Value
GET	/v1/tags	Get all tags

 © 2016 General Electric Company - All rights reserved. 225

The API documentation is available online on the predix.io site. In the Catalog, select the Time Series tile and scroll down to the API Documentation section.

- You can use the time series API to list tags and attributes, as well as to query data points. You can query data points using start time, end time, tag names, time range, measurement, and attributes.
- Use the **GET** method for less complex queries. Responses can be cached in proxy servers and browsers.
- Use the **POST** method as a convenience method when the query JSON is very long, and you do not need the request to be cached.



Creating a Query Header

- Method: POST or GET

To cache (GET) or not to cache (POST) if JSON is too long

- URL: https://query_url/v1/datapoints

- Authorization: Bearer token

- Predix-Zone-Id:

Zone-Id of Timeseries service instance

- Content-Type:

application/json

The screenshot shows a REST client interface with the following configuration:

- Method: POST (selected)
- Raw tab selected
- Headers section:
 - Add a new header
 - Authorization: Bearer oJSUiGti8j5Nh94fth5OvE
 - Content-type: JSON
 - Predix.Zone.Id: 644a721d-b85a-4094-bc17-f45k...



Creating a Query – Define Payload

Add query properties

- Start/End Time
- Data Points
- Order
- Aggregation
- Filter
- Groups

GET POST PUT PATCH DELETE

Raw Form

Add a new header

Authorization	Bearer oJSUiHGTi8j5Nh94fth5OvE
Content-type	JSON
Predix.Zone.Id	644a721d-b85a-4094-bc17-f45k...

```
{  
    "start": "15mi-ago",  
    "tags": [  
        {  
            "name": "ALT_SENSOR",  
            "limit": 1000  
        }  
    ]  
}
```



© 2016 General Electric Company - All rights reserved.

227



Sample Query

- Epoch 0 = 1970-01-01
- Specify start field – Absolute or Relative
- End field is optional – Absolute or Relative
If no time specified, query uses current date and time

Absolute time:

- Integer timestamp in milliseconds:

Relative time:

<value><time-unit>-ago:

- Name: This is the tag name – alphanumeric, periods, dashes, and underscores allowed
- Limit: Limit the number of data points returned in the query result

GET	POST	PUT	PATCH	DELETE
<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Raw	Form			
Add a new header				
Authorization	Bearer oJSUiHGTi8j5Nh94fth5OvE			
Content-type	JSON			
Predix.Zone.Id	644a721d-b85a-4094-bc17-f45k...			
<pre>"start": 1427463525000, "end": "5mi-ago", "tags": [{ "name": "ALT_SENSOR", "limit": 1000 }</pre>				



© 2016 General Electric Company - All rights reserved.

228

Start and End Time

Your query must specify a start field, which can be either absolute or relative. The end field is optional, and can be either absolute or relative. If you do not specify the end time, the query uses the current date and time.

- **Absolute start time** – Expressed as an integer timestamp value in milliseconds.
- **Relative start time** – Calculated relative to the current time. For example, if the start time is 15 minutes, the query returns all matching data points for the last 15 minutes. The relative start time is expressed as a string with the format <value><time-unit>-ago, where:
 - <value> must be an integer greater than zero (0).
 - <time-unit> – must be represented as follows:

Time Unit	Represents
ms	milliseconds
s	seconds
mi	minutes
h	hours
d	days
w	weeks
mm	months
y	years



Query Example - Data Points Options

- By Start and end field – Absolute or Relative
- Specify a limit for number of data points returned
- Ascending by timestamp
- Maximum limit of returns 200,000

GET POST PUT PATCH DELETE

Raw	Form
Add a new header	
Authorization	Bearer oJSUiHGTi8j5Nh94fth5OvE
Content-type	JSON
Predix.Zone.Id	644a721d-b85a-4094-bc17-f45k...

```
"start": 1427463525000,
"end": 1427483525000,
"tags": [
  {
    "name": "ALT_SENSOR",
    "order": "desc"
  }
]
```

Specifying the order of data points returned



© 2016 General Electric Company - All rights reserved.

229

When you create a query, you can **specify a limit for the number of data points returned** in the query results, as shown in the Query Properties example.

The default order for query results is **ascending**, and it is **ordered by timestamp**. You can use the order property in your query request to specify the order in which you want the data points returned. To return data in descending order use the order property:

Data points in the query **result cannot exceed the maximum limit of 200,000**. Narrow your query criteria (for example, time window) to return a fewer number of data points.



Data Points – Most Recent Data Point

- Filters are the only operation supported
- Specifying start and end times is optional
- If you specify a start time must also specify end time (and reverse)
- Default – query goes back 15 days

Example of query for latest data point with no time window defined

GET	POST	PUT	PATCH	DELETE
<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Raw	Form			
Add a new header				
Authorization	Bearer oJSUiHGTi8j5Nh94fth5OvE			
Content-type	JSON			
Predix.Zone.Id	644a721d-b85a-4094-bc17-f45k...			
<pre>"tags": [{ "name": "ALT_SENSOR", "filters": { "measurements": { "condition": "le", "value": 10 } } }]</pre>				



© 2016 General Electric Company - All rights reserved.

230

POST – Most Recent Data Point Rules

The Time Series API provides a POST method for retrieving the **most recent data point** within the defined time window. When using this API, the following rules apply:

- Filters are the only operation supported.
- Time parameters:
 - Specifying a start time is optional. However, if you do specify the start time, you must also specify an end time.
 - If you specify an end time, you must also specify the start time.
 - If you do not specify a start or end time, then by default, the query goes back 15 days and returns values through the current time.



Query Example - Data Aggregations

- Merge different data series
- Performed on all data points in query
- Processed in order specified in query
- <query url>/v1/aggregations returns all supported aggregations

Example of average aggregation

GET	POST	PUT	PATCH	DELETE
<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Raw	Form			
Add a new header				
Authorization	Bearer oJSUiHGTi8j5Nh94fth5Ove			
Content-type	JSON			
Predix.ZoneId	644a721d-b85a-4094-bc17-f45k...			
<pre>"start": 1427463525000, "end": 1427483525000, "tags": [{ "name": ["ALT_SENSOR", "TEMP_SENSOR"], "limit": 1000, "aggregations": [{ "type": "avg", "interval": "1h" }] }]</pre>				

© 2016 General Electric Company - All rights reserved.

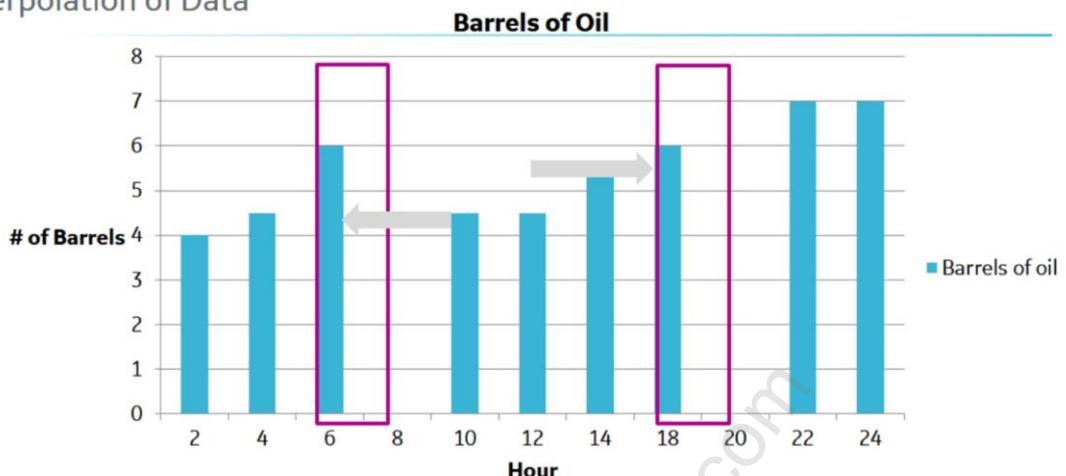
231



- Aggregation functions allow you to merge different data series into one single time series. Aggregation functions each perform distinct mathematical operations and are performed on all the data points included in the sampling period.
- See the documentation for a comprehensive list of aggregation types.
- You can use the <query url>/v1/aggregations endpoint to get a list of the available aggregations.
- Aggregations are processed in the order specified in the query. The output of an aggregation is passed to the input of the next until they are all processed.



Interpolation of Data



- Interpolate data between raw data samples when data is missing
- Query for results based on a time interval



© 2016 General Electric Company - All rights reserved.

232

You can interpolate data between raw data samples when data is missing and query for results based on a time interval (for example, every one hour).



Query Example – Filtering Data Points

Filter data points based on a condition

Operation	Attribute
Greater than (>)	gt
Greater than or equal to (\geq)	ge
Less than (<)	lt
Less than or equal to (\leq)	le
Equal to (=)	eq
Not equal to (\neq)	ne

Greater than or equal to operation with multiple tags

GET	POST	PUT	PATCH	DELETE
<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Raw	Form			
Add a new header				
Authorization	Bearer oJSUiHGTi8j5Nh94fth5OvE			
Content-type	JSON			
Predix.Zone.Id	644a721d-b85a-4094-bc17-f45k...			
<pre>"start": 1432671129000, "end": 1432671130500, "tags": [{ "name": ["tag119", "tag121"], "filters": ["measurements": { "condition": "ge", "values": "2" }] }]</pre>				



© 2016 General Electric Company - All rights reserved.

233

- Filter data points based on an **attribute value**
- Filters values for one or more tags
- Filters **qualities** for one or more tags



Grouping Data Points

Use the groups option in queries to return data points in specified groups:

- Attributes
- Quality
- Measurement
- Time



© 2016 General Electric Company - All rights reserved.

234

Use the groups option in queries to return data points in specified groups. Next, we will show you examples of grouping data points.



Grouping by Attributes

○ GET ○ POST ○ PUT ○ PATCH ○ DELETE

Raw Form

Add a new header

Authorization	Bearer oJSUihGti8j5Nh94fthSOVe
Content-type	JSON
Predix.Zone.Id	644a721d-b85a-4094-bc17-f45k...

```
{  
    "start":1432671128000,  
    "end":1432671129000,  
    "tags": [  
        {  
            "name": [  
                "tagname1",  
                "tagname2"  
            ]  
        }  
    ],  
    "groups": [  
        {  
            "name": "attribute",  
            "attributes": [  
                "attributename1"  
            ]  
        }  
    ]  
}
```

© 2016 General Electric Company - All rights reserved.

235



Grouping by Measurement

The rangeSize is the number of values to place in a group

○ GET ○ POST ○ PUT ○ PATCH ○ DELETE

Raw	Form
Add a new header	
Authorization	Bearer oJSUihGti8j5Nh94fthSOVe
Content-type	JSON
Predix.Zone.Id	644a721d-b85a-4094-bc17-f45k...

```
{  
  "start": "15d-ago",  
  "end": "1mi-ago",  
  "tags": [  
    {  
      "name": "tagname",  
      "groups": [  
        {  
          "name": "measurement",  
          "rangeSize": 3  
        }  
      ]  
    }  
  ]  
}
```



© 2016 General Electric Company - All rights reserved.

236

For example, a range size of 10 puts measurements between 0-9 in one group, 10-19 in the next group, and so on.



Grouping by Time

- Group data point results by a time range, beginning at the start time of the query
- The groupCount property defines the maximum number of groups to return

The return values are a range of one day, with one-hour groups

GET	POST	PUT	PATCH	DELETE
<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Raw	Form			
Add a new header				
Authorization	Bearer oJSUihGti8j5Nh94fthSOVe			
Content-type	JSON			
Predix.Zone.Id	644a721d-b85a-4094-bc17-f45k...			
<pre>{ "start": 1432671128000, "end": 1432671138000, "tags": [{ "name": ["tagname1", "tagname2"], "groups": [{ "name": "time", "rangeSize": "1h", "groupCount": 24 }] }] }</pre>				

© 2016 General Electric Company - All rights reserved.

237

The groupCount property defines the maximum number of groups to return. In the above example, with a rangeSize of "1h" and a groupCount of "24", the return values are a range of one day, with one-hour groups.



Use Case

Locomotive



© 2016 General Electric Company - All rights reserved.

238



Use Case: Locomotive

Goal	Solution implements Predix Services
<ul style="list-style-type: none"> • Improve fuel efficiency and lower costs • Capture data from locomotive sensors <ul style="list-style-type: none"> - RPM - Torque - Location (longitude, latitude) 	<ul style="list-style-type: none"> • Asset Data: define locomotive assets • Time Series: ingest and store sensor data • User Account and Authorization (UAA): authenticate clients • Views: display time series data in a dashboard • Redis: stores the user session



© 2016 General Electric Company - All rights reserved.

239

The next lab focuses on a transportation use case that illustrates how Predix services can be implemented in a client application. In this use case, our customer owns a fleet of locomotives distributed across the country. One of their KPIs is to improve the fuel efficiency of their fleets to lower overall costs of operation. To do this, they need to capture data emitted from sensors on each locomotive to track information such as fuel consumption, speed (RPM), torque and location (different terrain will effect fuel consumption).

You will build a client application using the following Predix services:
Time Series, Asset Data, UAA and Views.

In the classroom environment, you will use a microservice to generate (simulate) locomotive data, rather than connect directly to an edge device.



Use Case: Locomotive

Simulates the
data and calls
locomotive
data ingestion
app

**Locomotive
Simulator**

- **Data request**
- **Token request**
- **Token**
- **Data delivery**

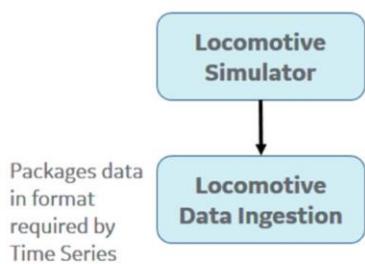


© 2016 General Electric Company - All rights reserved.

240



Use Case: Locomotive



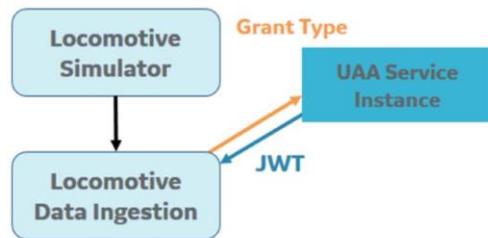
- **Data request**
- **Token request**
- **Token**
- **Data delivery**



© 2016 General Electric Company - All rights reserved.

241

Use Case: Locomotive



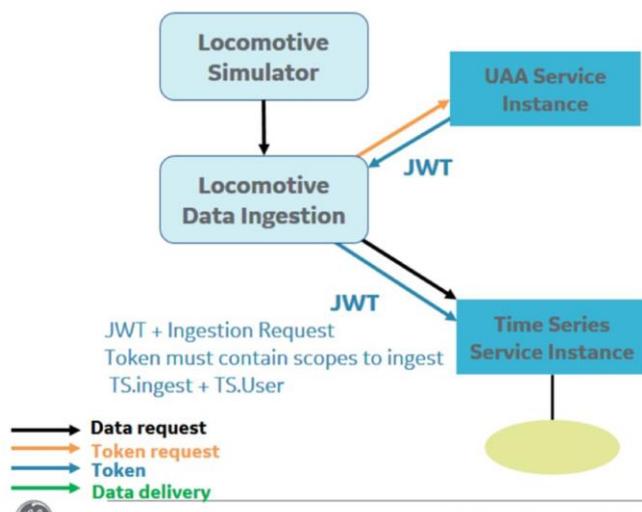
→ Data request
→ Token request
→ Token
→ Data delivery

© 2016 General Electric Company - All rights reserved.

242



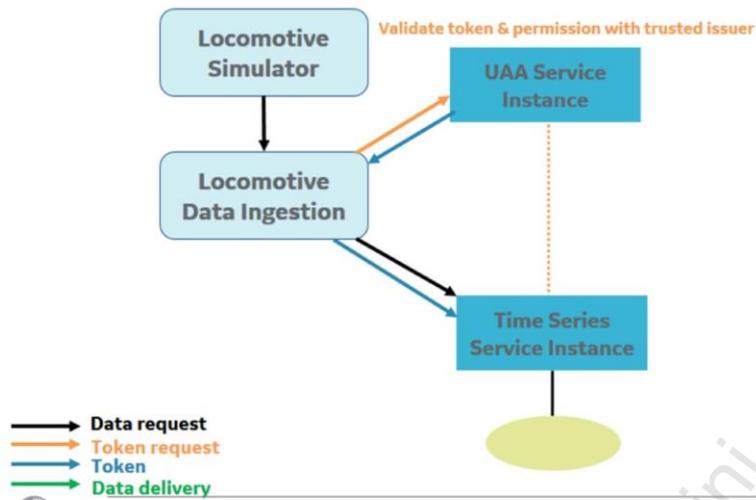
Use Case: Locomotive



243



Use Case: Locomotive



244



User Interface

Lab 6: Use Case: Locomotive



- Exercise 1: Create Time Series Service Instance
- Exercise 2: Build the Data Ingestion Application
- Exercise 3: Build the Simulator Application
- Exercise 4: Query Time Series Data



© 2016 General Electric Company - All rights reserved.

245

Exercise 1: The locomotive data ingestion application calls a time series service instance. Before you build the application, you will need to create a Time Series service instance.

Exercise 2: In this exercise you will build a data-ingestion service which will ingest data from the simulator service and push the data to the Timeseries database. The service has an endpoint /SaveTimeSeriesData which will accept an authorization token in the header, zoneId, clientId, and content JSON as the parameters. It will inject the JSON into the Timeseries database.

Exercise 3: In this exercise you will build a simulator service which will randomly generate 3 datasets every 2 seconds. The datasets consist of:

- RPM value
- Torque value
- Location tuple

Exercise 4: Use the Predix Starter Kit UI to query timeseries data generated by your locomotive-ingestion-service.



Security Setup

User needs to be part of these groups:

- views.zones.zoneid.user
- asset.zones.zoneid.user
- Timeseries.zones.zoneid.user
- Timeseries.zones.zoneid.query



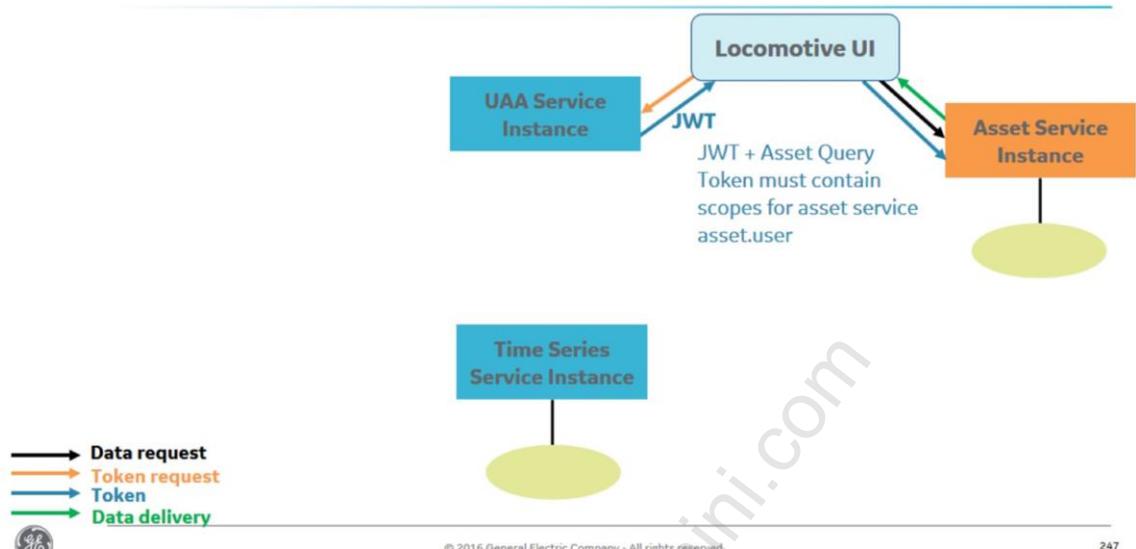
© 2016 General Electric Company - All rights reserved.

246

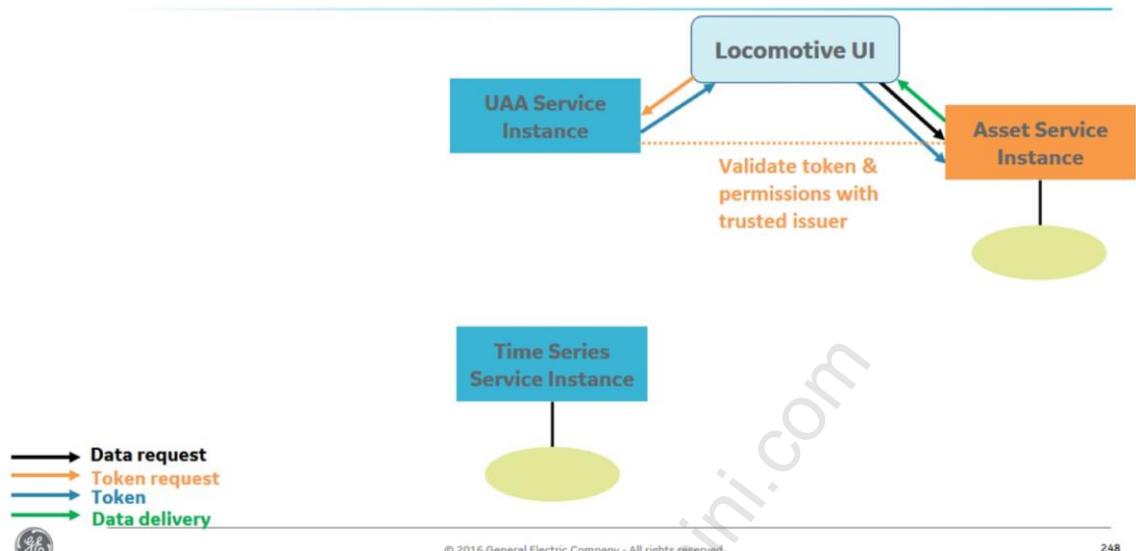
Client scopes field also requires same permissions.



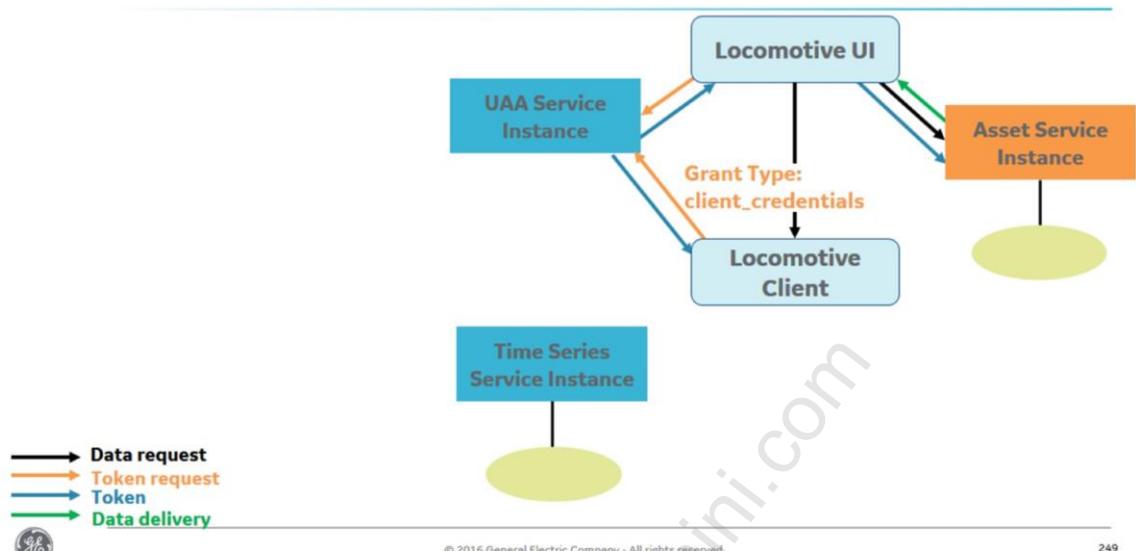
Use Case: Locomotive



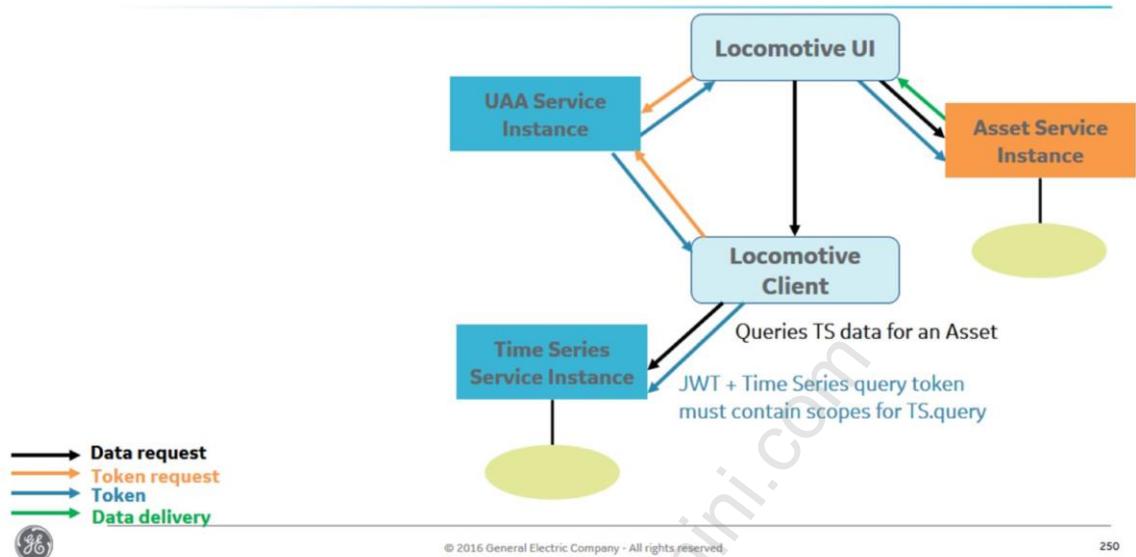
Use Case: Locomotive



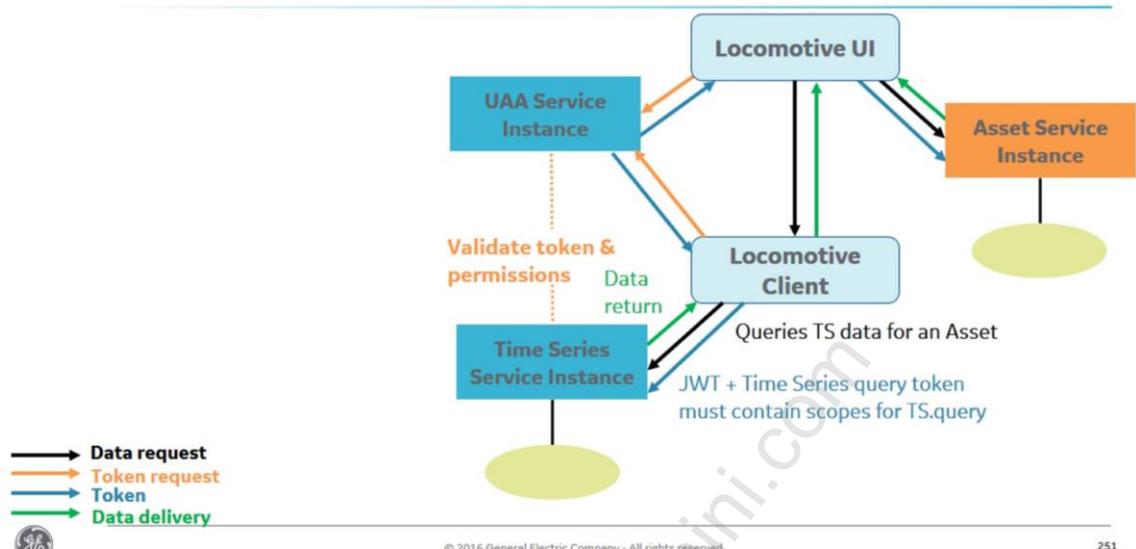
Use Case: Locomotive



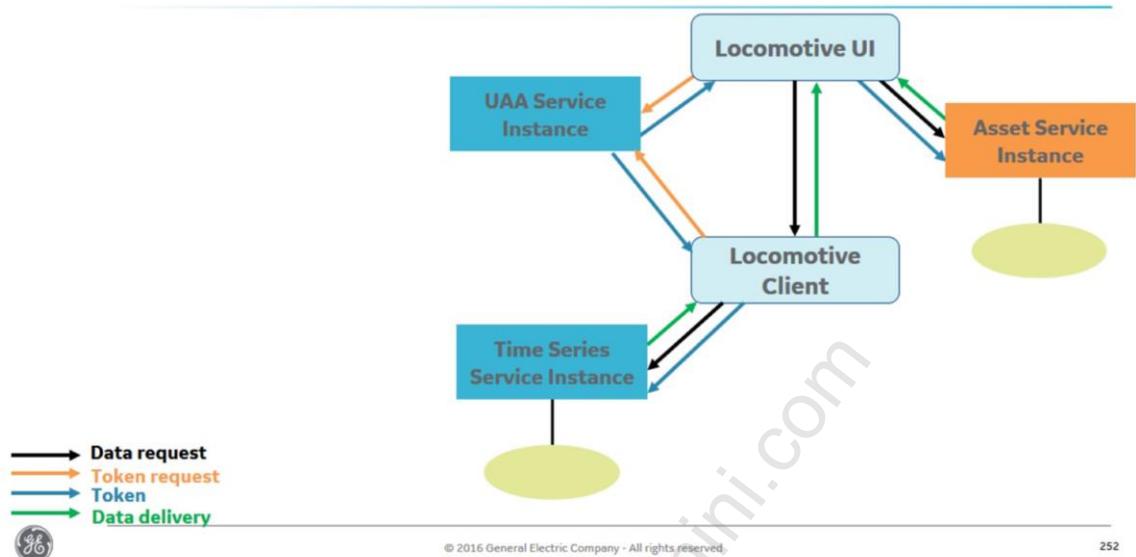
Use Case: Locomotive



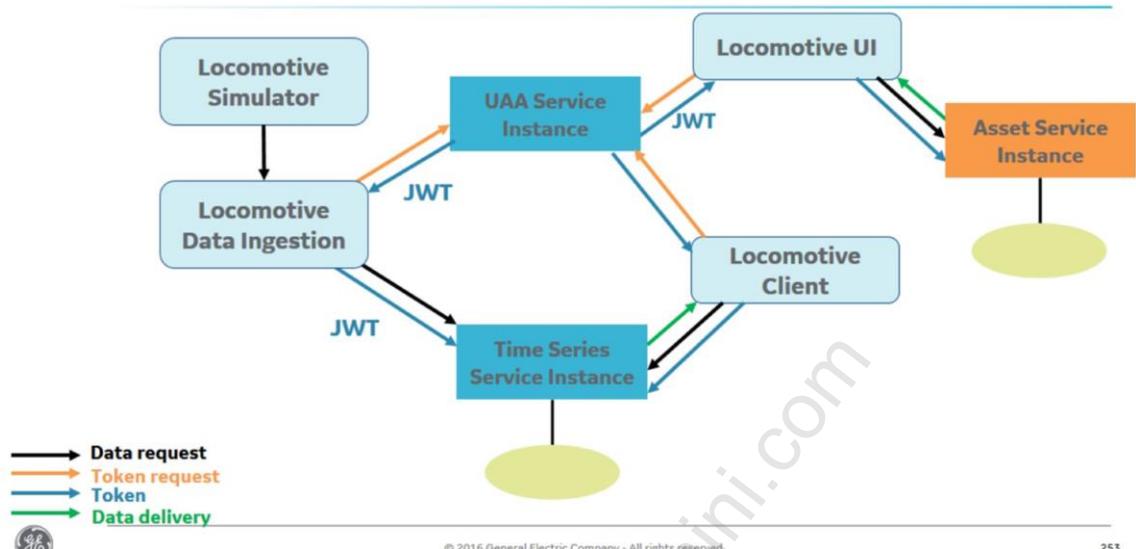
Use Case: Locomotive



Use Case: Locomotive



Use Case: Locomotive



User Interface

Lab 6: Use Case: Locomotive



- Exercise 5: Create a Locomotive Client Application
- Exercise 6: View Data in the Dashboard



© 2016 General Electric Company - All rights reserved.

254

Exercise 5: In the Asset lab, you loaded your data model into your asset service instance. As part of that activity, you added the asset OAuth scopes to your UAA client. These additional scopes were included in the UAA token, which allowed the client to access the asset API and post data to the asset database.

The Locomotive client application queries the timeseries data by calling the time series APIs.

Exercise 6: In this exercise you will build a dashboard app using the Predix Dashboard Seed. The dashboard will display the locomotive data stored in the timeseries database. The dashboard app calls the locomotive client API to retrieve the timeseries data. It also calls the asset service APIs to retrieve asset model data.



Module Summary

- Asset Data services provide REST APIs to support asset modeling
- Predix Asset service allows you to view & update asset models & retrieve asset data
- Query Asset data using GEL (Graph Based Data Model)
- Time Series has two components: Data Ingestion and Data Query



© 2016 General Electric Company - All rights reserved.

255



Working with Analytics



© 2016 General Electric Company - All rights reserved

256



Module Overview: Working with Analytics

**Analytics**

Overview	Use Case	Services Features		
Securing Analytics	UAA	UAA Env Variables	Lab 7	
Catalog Service	Catalog APIs	REST Calls	Manage Env Variables	Lab 7
Runtime Service	Running Orchestration	Scheduling a Job	Lab 7	
UI Service	UI Dashboard	Testing the Analytics	Lab 7	



© 2016 General Electric Company - All rights reserved.

258



Module Objectives: Working with Analytics

By the end of this module you will be able to:

- Examine the Predix Analytics services and explore their capabilities
- Learn how to upload and deploy analytics to the catalog
- Execute sample analytics
- Use web-based user interface to access the analytics

Completion of the following lab activities will reinforce the concepts covered in this module:

- Lab 7: Working with Analytics



© 2016 General Electric Company - All rights reserved

257



an·a·lyt·ics

Analytics is the scientific process of transforming data into insights for making better decisions.

Uses mathematics, statistics, predictive modeling and machine-learning techniques for:

- Discovery
- Interpretation
- Communication of meaningful patterns in data



© 2016 General Electric Company - All rights reserved

259

- noun
- The systematic computational analysis of data or statistics.
- Information resulting from the systematic analysis of data or statistics.

Analytics helps people question traditional ways of operation AND CAN LEAD TO POSITIVE CHANGE. It helps identify unproductive activities, redefine them, remove waste and optimize operations improving efficiency, productivity and profitability.

It can help predict future trends leading to better business / operations performance and decision making.



Use Case for Analytics

Train Performance	Analytics can help:	
<ul style="list-style-type: none">• Speed• Terrain (Grades)• Weather<ul style="list-style-type: none">- wind- Snow- temperature, etc.• Engine, wheel and track maintenance	<ul style="list-style-type: none">• Minimize Fuel Consumption• Optimize Productivity & Asset Utilization• Predict Future Performance• Maximization of Profitability	



© 2016 General Electric Company - All rights reserved

260

Consider a train :

- Fuel efficiency and consumption will depend on the speed, the terrain and weather conditions it is driven in
- Onboard sensors, meters and GPS produce a lot of relevant data that can be analyzed to find optimum train operating speed in any given weather or terrain condition to minimize fuel consumption
- Analytics can help find co-relations among fuel consumption, track/ wheel surface deterioration and speed/ terrain/ weather data. This can be used to predict future need for maintenance , on one hand eliminating un-necessary preventive maintenance and reducing maintenance/ downtime costs ; and on the other hand ensuring safe operation of the train
- This optimization will eventually lead to the maximization of profitability.



Predix Analytics Services

Analytics Runtime

Use this service to support elastic execution of the analytic orchestration.

PREDIX

Analytics User Interface

Use this browser-based user interface to upload, validate, and run analytics.

PREDIX

Analytics Catalog

Add analytics to the Predix cloud for use with the Analytics Runtime Service.

PREDIX



© 2016 General Electric Company - All rights reserved

261

Currently there are three Analytic services provided by Predix:

The first service, **The Analytics Catalog** provides a modular, hierarchical catalog for publishing and sharing reusable analytics across development teams.

The **Analytics Runtime** supports elastic execution of analytic orchestration

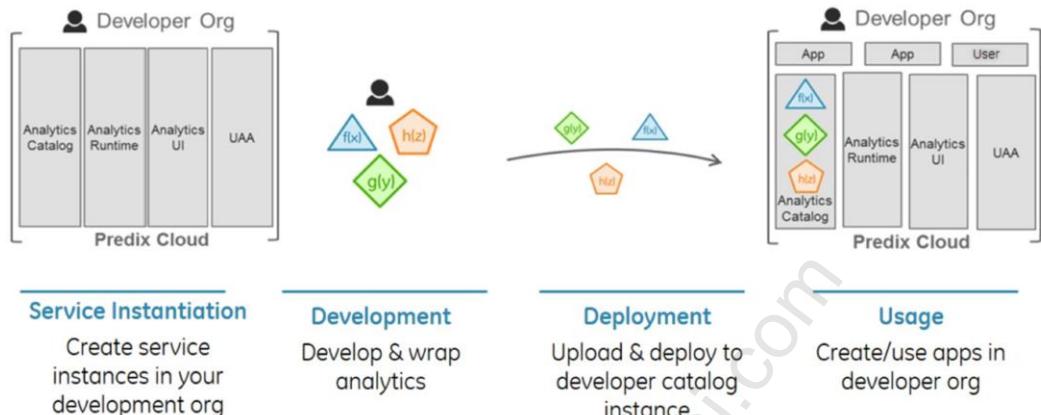
And the third service is **The Analytics User Interface**. This service allows data scientists to upload analytics written in Java, Python, or Matlab to the Analytics Catalog service.

When uploading an analytic, you can categorize the analytic in a hierarchical taxonomy and upload support files in addition to the analytic executable.

After an analytic is uploaded to the Analytics Catalog service, you can validate and perform test executions of the analytic by providing form-based input to see the results on a screen.



Analytics Services Workflow: Deploy and Use



© 2016 General Electric Company - All rights reserved

262



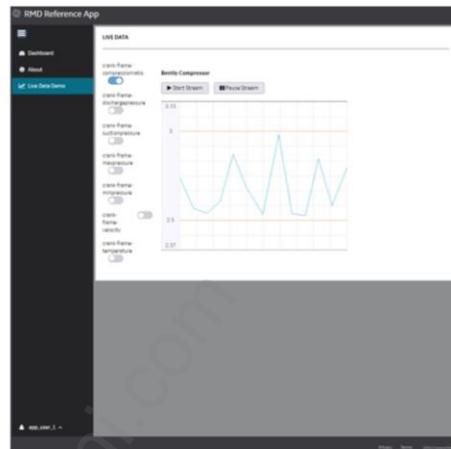
Analytics Dashboard

Data Visualization

Actionable Insights

Communicate asset and business performance metrics through visualizations

Fast decision-making and timely execution



© 2016 General Electric Company - All rights reserved

263

An important aspect of Analytics is not only to find meaningful data patterns in the vast quantities of data being generated but to communicate them in a highly effective manner in order to provide Actionable Insights.

Analytics Dashboard products communicate asset, operations and business performance through alerts, alarms, charts and other visualizations.

A central theme of GE's Industrial Internet strategy is optimizing both system and business performance. This distinction is useful when considering the appropriate KPIs and data visualizations to include in an Analytics Dashboard application and how to group them.

Asset performance might include information like utilization/maintenance summaries or safety alerts, or detailed performance metrics such as turbine start times or daily WIP of an asset.

Business performance might include information pertaining to a fleet's financials, such as operational expenditures or fleet-wide fuel efficiency.

In addition to business and asset distinctions, Analytics Dashboard applications may allow users to work at different levels of detail, from system-level overviews of groups of related analytics for high level managers to more detailed and rigorous visualizations exposing a single data type for engineers and technicians working with a specific asset.



Analytics Catalog Service - Features

- Repository for hosting analytics
 - Max file size for artifact upload- 250MB
 - Max expanded file size (including directories and files) - 500MB
- Taxonomy – hierarchical categorization of analytics
- Catalog REST APIs
 - Manage catalog entries through CRUD (create, retrieve, update, delete) operations
 - Deploy of analytics and version control
 - Upload and download artifacts, documentation, and test data
 - Analytic validation
 - Execute analytic



264

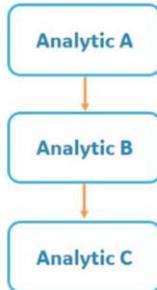
The Analytics Catalog service offers a repository for hosting analytics and exposing them as a service in the Predix platform:

- Each analytic entry in the Analytics Catalog contains an executable that will be deployed to the Predix platform
- If the artifact file being uploaded is too large, it may need to be chunked
- The entry may also contain documentation, test data, or other artifacts related to the analytic
- REST APIs are provided for managing analytics, publishing and sharing analytics with other users, and running analytics in the cloud



Analytics Runtime Service - Features

- Supports elastic execution of the analytic orchestration
- Orchestration – workflow grouping and sequencing of analytics



- Job scheduling

Analytics Runtime

Use this service to support elastic execution of the analytic orchestration.

PREDIX



The runtime service comprises of three components : Configuration, orchestration and job scheduling.

Once your analytic has been uploaded to the Catalog, validated, and deployed, you use the Runtime service to execute your analytic.

This is done by running an orchestration of one or more analytic.

Each orchestration defines the order or sequence for executing individual analytics.

The Analytic Runtime service provides a scheduler feature for scheduling an individual analytic or an time based orchestration.

You can also define how often jobs should run and specific start (**PAUSE**) or end times. The Predix Analytics Runtime REST APIs expose a mechanism for submitting BPMN orchestrations for execution.

Configuration - provide APIs configuration details such as the orchestration and data handling for each step in the orchestration.

You can perform analytic orchestration through the framework's configuration and parametrization capabilities, reducing the need for custom coding and point-to-point integrations.



Analytics User Interface Service - Features

Provides a web-based way to access Analytics Catalog and Runtime

- Add analytics to the catalog
- Deploy and test analytics
 - Enter inputs
- Delete analytics from the catalog
- Upload and download artifacts (executables, documentation, input/output files)
- Authorized access secured by UAA



© 2016 General Electric Company - All rights reserved.

266

The Analytics User Interface (UI) service provides web based access to Analytics Catalog and Analytics Runtime features.

It is a web application for uploading and testing your analytics. Common tasks such as adding an analytic to the catalog, deleting an analytic from the catalog, testing an analytic in a Cloud Foundry environment, and downloading an analytics' artifact can be completed in the UI.

Your UI instance will be multi-tenant aware. Only authorized users will be able to access your UI instance. Analytics Catalog and Runtime data accessed using the UI will be restricted based on the instance's tenant.



Analytics Development

Supported languages:

- Java – supports JDK 1.7+
- Matlab
 - as a Java module
 - consumes Matlab as a library
- Python



© 2016 General Electric Company - All rights reserved

267

- Analytics can be written in Java, Matlab or Python.
- Future support s planned for R, C# and C++.



Working with Analytics

Securing Your Analytics



© 2016 General Electric Company - All rights reserved

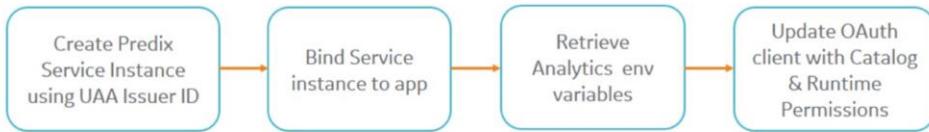
268





Update OAuth Client for Catalog and Runtime

- Establish service_scopes for Analytics services



All Predix microservices require a UAA instance to ensure secure use



© 2016 General Electric Company - All rights reserved

269

Here, we walk you through the steps required to update OAuth client for Catalog and Runtime services:

- Create a Predix Service instance, using the UAA Issuer ID, in this example we are creating the analytics Catalog and Runtime service
- Bind your service instance to your app
- Fetch your OAuth scope from the APP environment variables
- Update your OAuth client with Analytics Catalog and Analytics Runtime permissions (OAuth scopes and user authorizations)



Locating Analytics Catalog Environment Variables

Permissions to be added to OAuth client

```
}
"predix-analytics-catalog": [
  {
    "credentials": {
      "catalog_uri": "https://predix-analytics-catalog-release.run.aws-usw02-pr.ice.predix.io",
      "zone-http-header-name": "Predix-Zone-Id",
      "zone-http-header-value": "75354346-d024-4ac2-b143-f46886aedaad",
      "zone-oauth-scope": "analytics.zones.75354346-d024-4ac2-b143-f46886aedaad.user"
    },
    "label": "predix-analytics-catalog",
    "name": "vidya-analytics-catalog",
    "plan": "Bronze",
    "provider": null,
    "syslog_drain_url": null,
```



© 2016 General Electric Company - All rights reserved.

270

To enable application to access the Analytics Catalog and Runtime services, your JSON Web Token (JWT) must contain the following scope(s) for both the catalog and runtime services:

analytics.zones.<service_instance_guid>.user

Retrieve environment variables for catalog and runtime services



Locating Analytics Runtime Environment Variables

Permissions to be added to OAuth client

```
"predix-analytics-runtime": [  
  {  
    "credentials": {  
      "config_uri": "https://predix-analytics-config-release.run.aws-usw02-pr.ice.predix.io",  
      "execution_uri": "https://predix-analytics-execution-release.run.aws-usw02-pr.ice.predix.io"  
      "monitoring_uri": "https://predix-monitoring-service-release.run.aws-usw02-pr.ice.predix.io"  
      "scheduler_uri": "https://predix-scheduler-service-release.run.aws-usw02-pr.ice.predix.io",  
      "zone-http-header-name": "Predix-Zone-Id",  
      "zone-http-header-value": "05cad838-eb67-44c4-bba0-92f3a0758def",  
      "zone-oauth-scope": "analytics.zones.05cad838-eb67-44c4-bba0-92f3a0758def.user"
```



© 2016 General Electric Company - All rights reserved.

271

To enable application to access the Analytics Catalog and Runtime services, your JSON Web Token (JWT) must contain the following scope(s) for both the catalog and runtime services:

analytics.zones.<service_instance_guid>.user

Retrieve environment variables for catalog and runtime services.



Working with Analytics



Lab: 7 Working with Analytics

Part I: Your Dev Environment and UAA

- Exercise 1: Create and Bind an Analytics Catalog Service Instance
- Exercise 2: Create and Bind an Analytics Runtime Service Instance
- Exercise 3: Updating the OAuth2 Client



© 2016 General Electric Company - All rights reserved.

272

Exercise 1:

Before you begin this exercise, make sure that:

- An instance of the UAA service has been configured as your trusted issuer.
- An application has been bound to your UAA service instance

Exercise 2: You will follow similar steps to create the analytics runtime service instance.

Exercise 3: In this exercise you will update the OAuth2 client with the required scopes to work with the

Analytics Catalog and Runtime services.

`analytics.zones.<catalog_instance_guid>.user`

`analytics.zones.<runtime_instance_guid>.user`

You will then update the Runtime service instance with OAuth2 client credentials. This is required by the scheduler micro-service.



Analytics Services Lab Process Flow



Here is an overview of steps for setting up Analytic catalog service (Lab 7):

- Create a Catalog service instance using the trusted Issuer Id from your UAA service.
- Bind the analytic catalog instances to your Predix app.
- Once you have bound your catalog instance to your app, update the OAuth client authorities with your catalog instance OAuth scope. These scopes will be found in your app environmental variables.
- Fetch a UAA token using your client, use this token to call the analytics API's
- Use the REST client to call Analytic APIs.



Working with Analytics

Setting Up Services



© 2016 General Electric Company - All rights reserved

274



Catalog Service APIs

API Documentation

▶ Analytic Execution

▼ Analytics

GET	/api/v1/catalog/analytics	Get all analytic catalog entries.
POST	/api/v1/catalog/analytics	Create an analytic catalog entry.
GET	/api/v1/catalog/analytics/versions	Get all versions of the analytic catalog entry with the given name.
DELETE	/api/v1/catalog/analytics/{id}	Delete an analytic catalog entry by id.
GET	/api/v1/catalog/analytics/{id}	Get an analytic catalog entry by id.

predix.io/services – analytical catalog services – API documentation

 © 2016 General Electric Company - All rights reserved. 275

The following APIs are currently available:

Analytics - catalog entries and executions

- CRUD (Create, Retrieve, Update, Delete) analytics
- Deployment
- Validation

Artifacts

- Upload, download, update or delete an artifact
- Attach artifacts to a catalog entry

Taxonomy

- Load and retrieve taxonomy



Working with Analytics

Predix.io - APIs

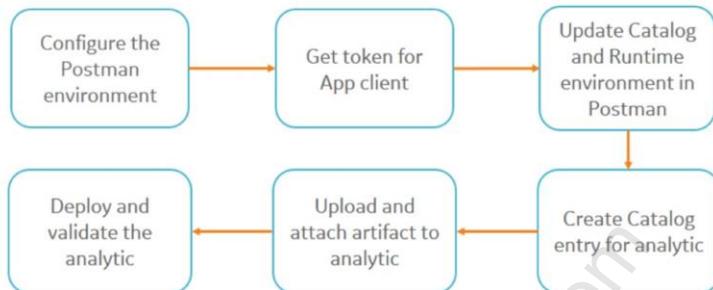


© 2016 General Electric Company - All rights reserved.

276



Working with Analytics Catalog APIs



Create and send any HTTP request using any REST client.

Steps to work with Analytics APIs:

- Import API collections into Postman and set the environment for Analytics.
- Get a UAA token using API, update the token value in your REST client environment.
- Update the environment tenant variables with your Catalog and Runtime service instance GUIDs. This sets up appropriate authorization and access for Analytics Catalog and Runtime
- Create an analytic catalog entry
- Upload an executable artifact (a .jar file) and attach it to the catalog entry
- Deploy and validate the analytic
- Retrieve validation status

Analytics can be developed in Python, Matlab or Java. Other languages such as R, C++ and C# will be supported in future.

Publish the algorithms to the catalog at a desired taxonomical location. Multiple versions of each analytic can be hosted and managed by the service.

The catalog enables sharing and reuse of modular analytics. The hierarchical catalog architecture for classifying and nesting analytics helps the user to easily locate analytics for a specific goal.

Here is an overview of steps in setting up Analytic catalog service:

1. Create a Catalog and Runtime service instances using the trusted Issuer Id from your UAA service
2. Bind the instances to your Predix app
3. Update the UAA app client authorities with your catalog and runtime service instance OAuth scope (**analytics.zones.<service_instance_guid>.user**)
4. Get UAA token using your client



5. Use REST client to call Analytics APIs

arun.moses@capgemini.com



Managing Environment Variables in Postman

Analytics-00 Student 20160201	
analyticId	a14dae0b-17a7-40a3-b0a8-b0dbe59
analyticName	Demo Adder Java 1
analyticVersion	v1
artifactId	cef31ec1-7bee-440f-8534-8bf62eb4l
bpmnXML_content	Value
catalog_tenant	0bfcc3b0-4626-4980-bc5b-cc3fb65
catalog_uri	predix-analytics-catalog-release.run.a
config_uri	predix-analytics-config-release.run.aw
configurationId	Value
Content-Type	application/json
deploymentrequestid	Value
execution_uri	predix-analytics-execution-release.rui
jobId	Value
protocol	https

Postman REST client allows for managing environment variables.

Postman REST client allows for managing environment variables.

Environment variables allow for more efficient API calls as they are used throughout the many API requests of the Analytics services. The values for the variables can be derived from the VCAP services environment (Analytics Catalog Zone ID, Runtime Zone ID, Time Series Zone ID, etc) or from the Analytics services responses (analyticId, analyticName , and arftifactId)



Managing Environment Variables in Postman

Analytics-00 Student 20160201			
<input checked="" type="checkbox"/> analyticId	a14dae0b-17a7-40a3-b0a8-b0dbe59	<input checked="" type="checkbox"/> jobId	Value
<input checked="" type="checkbox"/> analyticName	Demo Adder Java 1	<input checked="" type="checkbox"/> protocol	https
<input checked="" type="checkbox"/> analyticVersion	v1	<input checked="" type="checkbox"/> runtime_tenant	f76daftd-5cb0-475b-83ab-78dabec4f
<input checked="" type="checkbox"/> artifactId	cef31ec1-7bee-440f-8534-8bf62eb4l	<input checked="" type="checkbox"/> scheduler_uri	predix-scheduler-service-release.run:
<input checked="" type="checkbox"/> bpmnXML_content	Value	<input checked="" type="checkbox"/> token	eyJhbGciOiJSUzI1NiJ9eyJqdGkiOiJj
<input checked="" type="checkbox"/> catalog_tenant	0bfcc3b0-4626-4980-bc5b-cc3fb65	<input checked="" type="checkbox"/> UAA Token URL	a97db685-e3f5-4f10-ad03-f82bee5a
<input checked="" type="checkbox"/> catalog_uri	predix-analytics-catalog-release.run.a	<input checked="" type="checkbox"/> validationrequestId	34479f10-0a5b-4dd0-b711-53b0428
<input checked="" type="checkbox"/> config_uri	predix-analytics-config-release.run.aw	<input checked="" type="checkbox"/> X-Identity-Zone-Id	a97db685-e3f5-4f10-ad03-f82bee5a
<input checked="" type="checkbox"/> configurationId	Value		
<input checked="" type="checkbox"/> Content-Type	application/json		
<input checked="" type="checkbox"/> deploymentrequestId	Value		
<input checked="" type="checkbox"/> execution_uri	predix-analytics-execution-release.rui		
<input checked="" type="checkbox"/> jobId	Value		
<input checked="" type="checkbox"/> protocol	https		

Postman REST client allows for managing environment variables.

Postman REST client allows for managing environment variables.

Environment variables allow for more efficient API calls as they are used throughout the many API requests of the Analytics services. The values for the variables can be derived from the VCAP services environment (Analytics Catalog Zone ID, Runtime Zone ID, Time Series Zone ID, etc) or from the Analytics services responses (analyticId, analyticName , and arftifactId)



Obtain a Token

Set and update REST client environment

Variable values are referenced from the environment

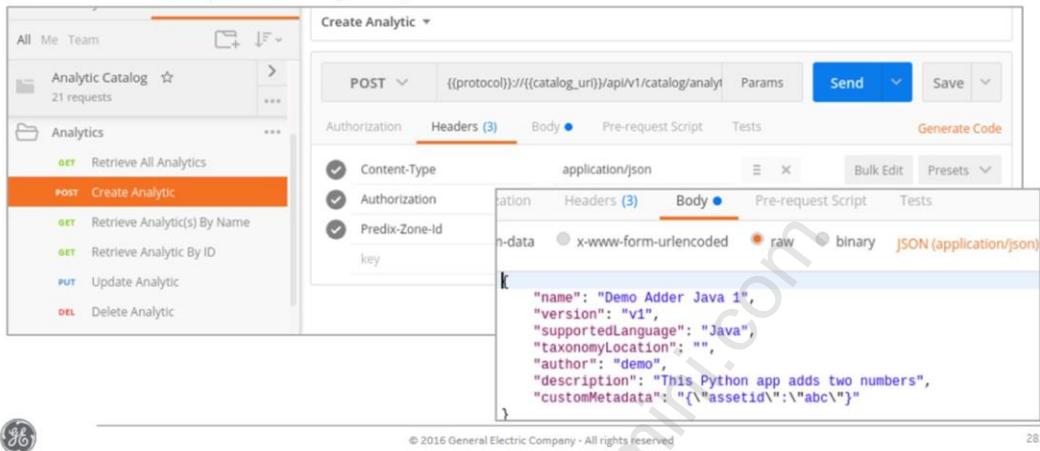
- This example shows a REST call to get a UAA token
- We are using the Postman REST client here



Uploading an Analytic the Catalog

Two step process:

1. Create a Analytic Catalog entry



The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** {{(protocol)}}://{{(catalog_uri)}}/api/v1/catalog/analytic
- Headers:**
 - Content-Type: application/json
 - Authorization: (JWT Token)
 - Predix-Zone-Id: key
- Body:** JSON (application/json) containing the following JSON object:


```

{
  "name": "Demo Adder Java 1",
  "version": "v1",
  "supportedLanguage": "Java",
  "taxonomyLocation": "",
  "author": "demo",
  "description": "This Python app adds two numbers",
  "customMetadata": "{\"assetid\":\"abc\"}"
}
      
```

Uploading an analytic to the catalog is a two step process:

First, you will need to POST a create analytic request in the catalog.

Remember to add Authorization and Predix zone ID headers and update the body with catalog entry information. This includes language, name, version, and description.

The Authorization header contains the JWT Token.

A unique analytic ID will be part of the response. This analytic ID can be used to attach artifacts to the catalog entry.



Uploading an Analytic the Catalog

Two step process:

2. Upload and attach an artifact to the entry

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Analytics' and 'Artifacts' sections, and a central panel with a 'Create Artifact' button. The main area is a POST request configuration:

- Method:** POST
- URL:** {{protocol}}://{{catalog_url}}/api/v1/catalog/artifacts
- Body:** (highlighted)
- Form Data:** (selected)
 - file: Choose Files (No file chosen)
 - catalogEntryId: {{(analyticId)}}
 - type: Executable
 - description: This analytic adds 2 numbers and pro...

Uploading an analytic to the catalog is a two step process:

First, you will need to POST a create analytic request in the catalog.

Remember to add Authorization and Predix zone ID headers and update the body with catalog entry information. This includes language, name, version, and description.

The Authorization header contains the JWT Token.

A unique analytic ID will be part of the response. This analytic ID can be used to attach artifacts to the catalog entry.



Working with Analytics



Lab: 7 Working with Analytics

Part II: Working with the Analytics Catalog through a REST Client

- Exercise 1: Getting Your UAA Token
- Exercise 2: Working with the Analytics Catalog



© 2016 General Electric Company - All rights reserved.

283

Objectives

By the end of this lab, students will be able to:

- Configure and use Postman (a REST client) to work with the API
- Set up appropriate authorization and access for Analytics Catalog and Runtime
- Create a catalog entry, upload an analytic executable, test and validate the analytic

Exercise 1: In this exercise, you will be using Postman to:

- Configure HTTP requests to your UAA service instance
- Retrieve UAA token

Exercise 2: For this lab exercise, you will work with a simple Analytic application demo, upload it to the catalog, then deploy and test it. The application adds two numbers together.

The following steps describe the process for adding a new analytic to the catalog:

- Create an analytic catalog entry
- Upload and attach an analytic artifact to the catalog entry
- Deploy and test the analytic in the Cloud Foundry environment



Analytics Services Lab Process Flow



Here is an overview of steps for setting up Analytic catalog service (Lab 7):

- Create a Catalog service instance using the trusted Issuer Id from your UAA service.
- Bind the analytic catalog instances to your Predix app.
- Once you have bound your catalog instance to your app, update the OAuth client authorities with your catalog instance OAuth scope. These scopes will be found in your app environmental variables.
- Fetch a UAA token using your client, use this token to call the analytics API's
- Use the REST client to call Analytic APIs.



Working with Analytics

Analytics Runtime Service

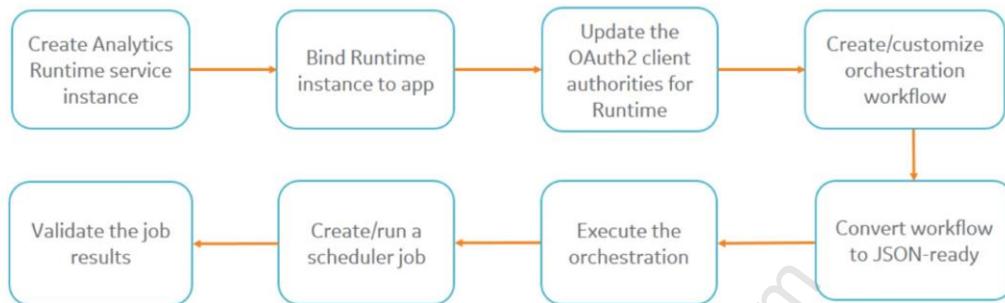


© 2016 General Electric Company - All rights reserved

285



Working with Analytics Runtime Service



© 2016 General Electric Company - All rights reserved

286

One of the functions of the Analytics runtime service is Orchestration.

Orchestration defines the order of analytic execution and supports the BPMN or Business Process Modeling Notation 2.0 standard.

Things to remember about orchestrations:

An orchestration execution request can be used to trigger multiple executions of an orchestration. For example, you can have one orchestration for each asset in a specified group of assets. You cannot run an analytic directly from a runtime service – even a single analytic must be defined as part of an **orchestration**.

With the scheduler function, you are able to run an orchestration at time-based intervals.

Lets review the Workflow the analytics runtime service:

- Retrieve your analytic info (version, name, and Id) using a GET request
- Download the sample BPMN workflow (OrchestrationWithOneAnalytic.xml) for the activity workflow engine from Github from the following address:
<https://github.com/PredixDev/predix-analytics-sample/tree/master/orchestrations>
- Customize the sample BPMN XML workflow with your analytic info. (AnalyticsId, Analytics Name, and Version).
- Convert the workflow to a JSON ready quoted string using 'Escape for JSON' tool
- Run the orchestration API request with the JSON ready BPMNxml
- Update the Postman environment with your Runtime instance scheduler_uri
- Create and schedule a job to execute orchestration, note the jobID
- Retrieve all jobs to see your scheduled job running
- Use the Get method to view execution results. This will show the job running at your specified time interval.



Retrieving Analytic Info

Use GET request to find analytic version, name, and id

Customize BPMN workflow and update REST environment variables

- Referenced by job scheduler while scheduling and running the job



```

Analytic Catalog  ☆  >
21 requests    ...
Analytics      ...
  GET Retrieve All Analytics
  POST Create Analytic
  GET Retrieve Analytic(s) ...
  GET Retrieve Analytic B...
  PUT Update Analytic
  ...
Delete Analytic
  
```

```

1  {
2   "analyticCatalogEntries": [
3     {
4       "author": "demo",
5       "supportedLanguage": "java",
6       "taxonomyLocation": "/uncategorized",
7       "customMetadata": "{\"assetid\":\"abc\"}",
8       "createdTimestamp": "2016-09-14T23:13:36+00:00",
9       "updatedTimestamp": "2016-09-14T23:13:36+00:00",
10      "description": "This Java app adds two numbers",
11      "version": "v1",
12      "name": "Demo Adder Java 1",
13      "id": "9e5923ca-fb3a-4703-92af-5cb46de26c43",
14      "state": "VALIDATED"
15    },
  
```

© 2016 General Electric Company - All rights reserved. 287

- Use GET request to find analytic version, name, and id and customize the sample BPMN workflow with these values
- Update the REST environment variables with these values and they will be referenced by job scheduler while scheduling and running the job



Running the Orchestration

The screenshot shows the Postman application interface. On the left, the 'Collections' sidebar lists several items: 'Analytic Catalog' (21 requests), 'Orchestration Configuration' (9 requests), 'Orchestration Examples' (2 requests), 'Run (Using environment ...)' (POST), and 'Run (Using sample bpmn...)' (POST). The 'Run (Using sample bpmn...)' item is highlighted with a red box. The main workspace is titled 'Run (Using sample bpmnXML)'. It shows a POST request to '{{protocol}}://{{execution_uri}}/api/v1/execution'. The 'Body' tab is selected, showing JSON input. The JSON content is a BPMN XML snippet:

```

1  {
2    "id": "Execution of Orchestration with One Analytic",
3    "name": "Orchestration with One Analytic",
4    "bpmnXml": "<?xml version='1.0' encoding='UTF-8'?><n<definitions xmlns='http://www.omg.org/spec/BPMN/20100524/MODEL'><n      xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'><n          expressionLanguage='http://www.w3.org/1999/XPath' id='sid-81430087-7a44-4be3-8517-914faf923256'><n              targetNamespace='DSP-PM' typeLanguage='http://www.w3.org/2001/XMLSchema'><n                  xsi:schemaLocation='http://www.omg.org/spec/BPMN/20100524/MODEL http://www.omg.org/spec/BPMN/2.0/20100501/BPMN20.xsd'><n                      xmlns:activiti='http://activiti.org/bpmn'><n                          <process id='OrchestrationWithOneAnalytic' isExecutable='true'><n                              <startEvent id='sid-start-event' name=' '><n                                  <outgoing>sid-flow1</outgoing><n                                  <serviceTask id='sid-10001' name='5f5f869d-2d29-422f-b0f3' startQuantity='1'><n                                      <!--><n                              </process><n</definitions>
  
```

Run the orchestration API request with the JSON ready BPMNxml. It will return a status of “200 Ok” if successful.



Creating a Scheduler Job

- Retrieve Scheduler Service uri from environment variables
- Update the REST environment to reflect the uri

```
"predix-analytics-runtime": [
  {
    "credentials": {
      "config_uri": "https://predix-analytics-config-release.run.aws-usw02-pr.ice.predix.io",
      "execution_uri": "https://predix-analytics-execution-release.run.aws-usw02-pr.ice.predix.io",
      "monitoring_uri": "https://predix-monitoring-service-release.run.aws-usw02-pr.ice.predix.io",
      "scheduler_uri": "https://predix-scheduler-service-release.run.aws-usw02-pr.ice.predix.io",
      "zone-http-header-name": "Predix-Zone-Id",
      "zone-http-header-value": "05cad838-eb67-44c4-bba0-92f3a0758def",
      "zone-oauth-scope": "analytics.zones.05cad838-eb67-44c4-bba0-92f3a0758def.user"
    }
  }
]
```



Creating and Verifying a Job Schedule

- Create a job to run orchestration at specified time interval
- Call job history request to verify execution

The screenshot shows the Predix Scheduler Service interface. On the left, there's a sidebar with 'History' and 'Collections'. Under 'Collections', 'Scheduler Service' is selected, showing 7 requests. Below the sidebar is a list of API endpoints:

- GET Retrieve All jobs**
- PUT Create Job**
- POST Create Job 2**
- GET Get Job By ID**
- PUT Update job**
- DEL Delete job**
- GET Get Job History By job ID**

Two specific job entries are highlighted with red boxes and annotations:

```

{
    "httpMethod": "POST",
    "cron": "0 0/1 * * * ? *",
    "httpStatusCode": 200,
    "jobId": "a018cff8-6aeb-4fd2-8543-a5a52c35e557",
    "scheduledFireTime": "2016-08-15 23:27:00.0",
    "fireTime": "2016-08-15 23:27:00.02",
    "timeZoneId": "UTC",
    "statusMessage": "completed",
    "url": "https://5f5f869d-2d29-422f-b0f3-0032469515ab.run.aws-usw02-pr.ice.predix.io/api/v1/analytic/execution",
    "id": "5f3be9fa-0de4-477b-b73c-8e0eee97b1e5",
    "result": "{\"result\":30}"
},
{
    "httpMethod": "POST",
    "cron": "0 0/1 * * * ? *",
    "httpStatusCode": 200,
    "jobId": "a018cff8-6aeb-4fd2-8543-a5a52c35e557",
    "scheduledFireTime": "2016-08-15 23:26:00.0",
    "fireTime": "2016-08-15 23:26:00.11",
    "timeZoneId": "UTC",
    "statusMessage": "completed"
}
  
```

A callout bubble points to the first highlighted entry with the text "Job running every minute".

1. Update the REST client environment with your Runtime instance scheduler_uri.
2. Create and schedule a job to execute orchestration, note the jobID.
3. Retrieve all jobs to see your scheduled job running.
4. Get the execution results by using a 'GET job history by jobId' request. This will show the job running at your specified time interval (every 2 min in this case).



Scheduling a Job

- Use the Post request to create a job
- It will return status “201 Created” denoting successful job creation
- Note the Job ID returned in the body of response and update your Postman environment with the Job Id

© 2016 General Electric Company - All rights reserved

291

Scheduling the execution of an orchestration or an individual analytic can be done on time-based intervals.

REST APIs are provided for managing a scheduled analytic or orchestration execution (also called a job)

Here, we are scheduling the demo adder analytic we created before to run once every minute, the execution request will point to the analytic catalog Id of this analytic.

We also provide 2 numbers as input to the analytic

```
executionRequest url": "<your analyticid>.run.aws-usw02-pr.ice.predix.io/api/v1/analytic/execution"
"value": "<your_catalog_instance_Zone-ID>"
"inputData": "{\"number1":10,\"number2\":20}"
```



Working with Analytics



Lab: 7 Working with Analytics

Part III: Working with the Runtime service

- Exercise 1: Executing an Analytic with Runtime service
- Exercise 2: Schedule an Orchestration and Analytic execution



© 2016 General Electric Company - All rights reserved.

292

Learning Objectives

By the end of this lab, students will be able to:

- Understand and create an orchestration of an analytic
- Create, test and validate a job schedule for an analytic

Lab Exercises

Exercise 1: You will use the following information, from the previous lab, to run an orchestration:

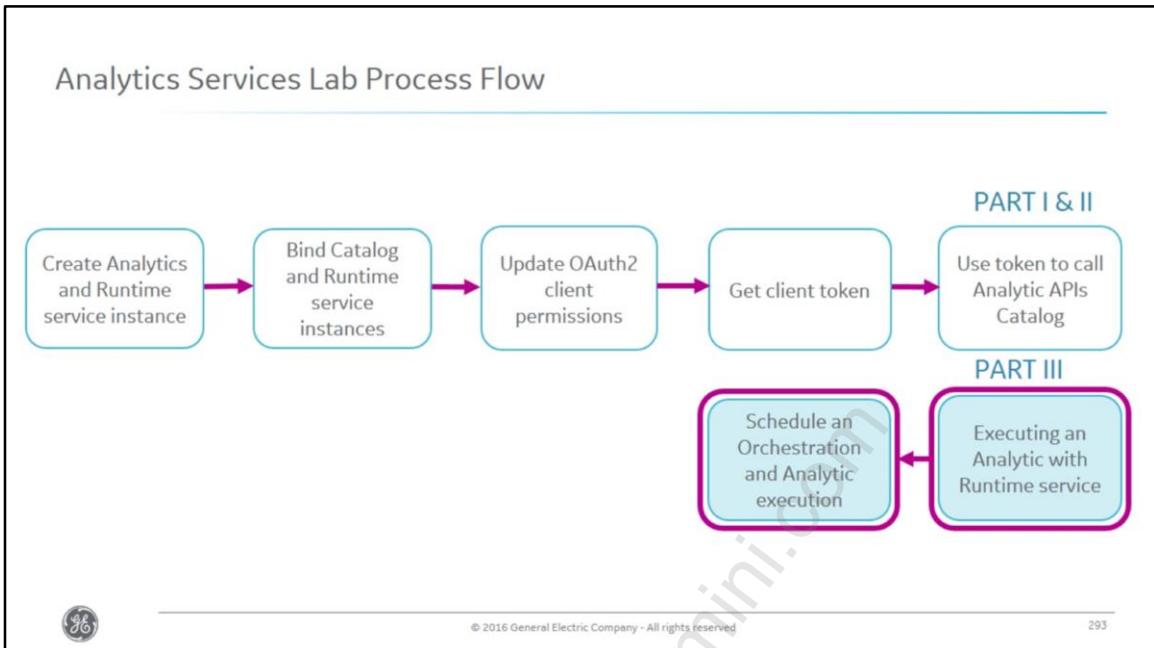
- Analytic catalog ID
- Analytic name
- Analytic version

The following exercise walks you through the process for running an orchestration with one analytic.

Exercise 2: Scheduling the execution of an orchestration or an individual analytic can be done on time-based intervals. REST APIs are provided for managing a scheduled orchestration execution:

- Create a job definition
- Retrieve job definitions
- Retrieve job history
- Update job definitions
- Delete existing jobs





Here is an overview of steps for setting up Analytic catalog service (Lab 7):

- Create a Catalog service instance using the trusted Issuer Id from your UAA service.
- Bind the analytic catalog instances to your Predix app.
- Once you have bound your catalog instance to your app, update the OAuth client authorities with your catalog instance OAuth scope. These scopes will be found in your app environmental variables.
- Fetch a UAA token using your client, use this token to call the analytics API's
- Use the REST client to call Analytic APIs.



Working with Analytics

Analytics UI Service



© 2016 General Electric Company - All rights reserved

294



Analytics UI Setup

Services Setup

- UAA
- Analytics Catalog
- Analytics Runtime

Security Setup

- Grant Type - authorization_code, refresh_token, client_credentials
- Client scopes – Runtime and Catalog OAuth scopes
- Client Authorities – Runtime and Catalog OAuth Scopes
- Create Runtime and Catalog groups
- Add User to Runtime and Catalog groups



© 2016 General Electric Company - All rights reserved.

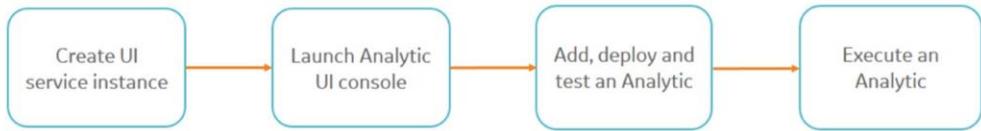
295

- You must have set up the **UAA**, **Analytics Catalog**, and **Analytics Runtime** service instances the UI will interact with before creating UI service instance
- Authentication access to analytics UI services is controlled by the designated trusted issuer and is managed by the User Account and Authentication (UAA) web service
- One UI service instance can manage a single UAA, Analytics Catalog, and Analytics Runtime service combination
- Several configuration parameters must be set to identify the dependent service instances



Working with Analytics UI Service

- Provides web based access to Analytics features
- Only authorized users access your UI



Here we will review the steps to create your Analytics UI Service.

First, you will need to create the UI service instance using UAA service instance uri, analytics runtime GUID, and catalog service instance GUID. You will also need to create a Client ID, client secret, and a domain name.

Next launch the Analytics UI console in a web browser using the URL for your UI service instance. Finally, Add, deploy, test, and execute an analytic.



Creating the Analytics UI Service Instance Using Predix.io

To create a UI instance using Predix.io you need:

- Your organization and space
- UAA select you UAA instance from the drop down list
- Service Instance name
- Service plan associated with the service
- Client_id
- Client secret

The screenshot shows the 'New service instance' creation page in the Predix Developer portal. It includes fields for Org, Space, User Account & Authentication (UAA), Service instance name, Service plan, Client id, Client secret, and Domain prefix. A note at the top states: 'The service you are trying to subscribe to requires User Account & Authentication'.



© 2016 General Electric Company - All rights reserved

297

To create a UI instance using Predix.io, you will need the following info:

- Your organization and space
- UAA select you UAA instance from the drop down list
- Service Instance name is the name of your service instance
- Service plan is the plan (Free) associated with the service
- Client_id is the client id to be used with your UAA service instance
- Client secret is the client secret to be used with your UAA service instance
- Domain_prefix is the domain prefix to be used to access your Analytics UI in a browser. It is case insensitive. For example, assume the base Predix Analytics User Interface URL is predix-analytics-ui.run.aws-usw02-pr.ice.predix.io. If you provide the domain name "abc" at creation, then your user interface instance URL is abc.predix-analytics-ui.run.aws-usw02-pr.ice.predix.io
- Catalog predix zone id is the service instance GUID of your Analytics Catalog instance (cf service <my-analytics-catalog-service> --guid)
- Runtime predix zone id is the service instance GUID of your Analytics Runtime instance (cf service <my-analytics-runtime-service> --guid)

The message on the screen confirms that the service instance has been created.



Get Your UI Dashboard URL

`cf service <UI service instance>`



```
[predix@localhost ~]$ cf service predix-analytics-ui-gr
Service instance: predix-analytics-ui-gr
Service: predix-analytics-ui
Plan: Free
Description: Use this browser-based user interface to upload, validate, and run analytics.
Documentation url:
Dashboard: https://training-gr.predix-analytics-ui.run.aws-usw02-pr.ice.predix.io

Last Operation
Status: create succeeded
Message:
Started: 2016-03-04T17:10:29Z
Updated: 2016-03-04T17:10:29Z
[predix@localhost ~]$
```



© 2016 General Electric Company - All rights reserved.

298

From the terminal, run `cf service <UI service instance>` to get the UI dashboard URL.



View the Analytics Catalog Dashboard

- Use the UI dashboard uri in a browser to get to login screen
- Login with the username and password

The screenshot shows a table with columns: Name, Version, Taxonomy Location, Author, and Description. Two rows are visible: 'Demo Adder Java 1' (v1, /uncategorized, demo, 'This Python app adds two...') and 'Shift Anomaly Detection' (V1-Mar-17, /Anomaly Detection, GE Predix Data Science ..., 'A univariate anomaly de...'). A blue box highlights the 'Delete analytic' button next to the second row.

Name	Version	Taxonomy Location	Author	Description
Demo Adder Java 1	v1	/uncategorized	demo	This Python app adds two...
Shift Anomaly Detection	V1-Mar-17	/Anomaly Detection	GE Predix Data Science ...	A univariate anomaly de...

The home page of the UI service will be the Analytics catalog listing all analytics available. You can add, update and delete analytics using this dashboard.

- To add analytic:
- Click on the 'New Analytic' button
- Select a Java, Matlab, or Python executable file available
- Provide meta data

You can download an artifact previously uploaded with an analytic from the Analytic Detail page.

Note: Maximum file size for upload is 250MB. The maximum expanded analytic file size (including directories and files) is 500MB.



Deploying and Testing the Analytic

The screenshot shows a user interface for deploying and testing an analytic. At the top, there is a navigation bar with four tabs: 'Deploy and Test' (highlighted with an orange border), 'Test', 'Release', and 'Logs'. Below the navigation bar, there are two main sections: 'Analytic Input' and 'Analytic Output'. In the 'Analytic Input' section, step 1 shows the tab 'Deploy and Test' highlighted. Step 2 shows the input field containing the JSON object '{ "number1":23, "number2":29 }'. Step 3 shows the button 'Submit for Deployment and Test'. In the 'Analytic Output' section, step 4 shows the status 'COMPLETED'. A note below the status says: 'Note: It may take a few minutes to return a status complete with your output result.' The output field displays the JSON object '{ "result":52 }'. At the bottom of the interface, there is a copyright notice: '© 2016 General Electric Company - All rights reserved' and a page number '300'.

To deploy and test your analytic app, navigate to the deploy and test lab.

Next, enter the input for your equation.

Once your input is entered, click 'Submit for Deployment and Test'.

It may take a few minutes for the application to process your request and return a complete status,



Working with Analytics



Lab 7: Analytic UI Service

Part IV: Using the Analytics User Interface

- Exercise 1: Update OAuth2 Client and Create UI User
- Exercise 2: Create Your Analytics UI Service Instance
- Exercise 3: Using the Analytics UI



© 2016 General Electric Company - All rights reserved.

301

Overview

Before using the Analytics UI, you will need to update the OAuth2 Client and then create a user that can log in and use the UI. The following exercise walks you through the process for running an orchestration with one analytic.

Exercise 1: Update OAuth2 Client and Create UI User

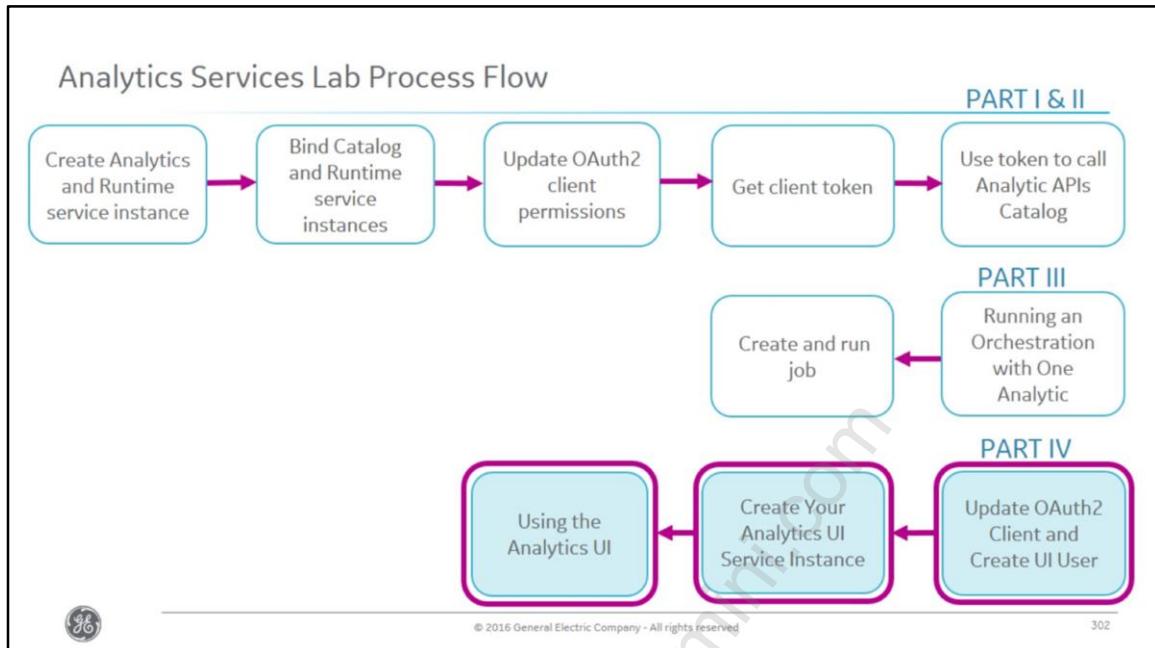
Exercise 2: Create Your Analytics UI Service Instance

There are two ways you can create an Analytics UI service instance: using cloud foundry (CF) CLI or using the Predix.io web interface (detailed steps in the Appendix).

Exercise 3: Using the Analytics UI

Once the UAA user and the UI service instance is created you will be able to use the UI to work with the catalog, uploading, deploying and testing your analytic applications.





Here is an overview of steps for setting up Analytic catalog service (Lab 7):

- Create a Catalog service instance using the trusted Issuer Id from your UAA service.
- Bind the analytic catalog instances to your Predix app.
- Once you have bound your catalog instance to your app, update the OAuth client authorities with your catalog instance OAuth scope. These scopes will be found in your app environmental variables.
- Fetch a UAA token using your client, use this token to call the analytics API's
- Use the REST client to call Analytic APIs.



Module Summary: Analytics

- Analytics can be developed using Java and Python
- Catalog service can be used to upload, execute and validate analytics
- Use Runtime service to create and run orchestrations
- Analytic catalog and runtime services require UAA authentication
- Analytics UI requires UAA, Catalog, and Runtime services configurations



© 2016 General Electric Company - All rights reserved

303



Predix Machine

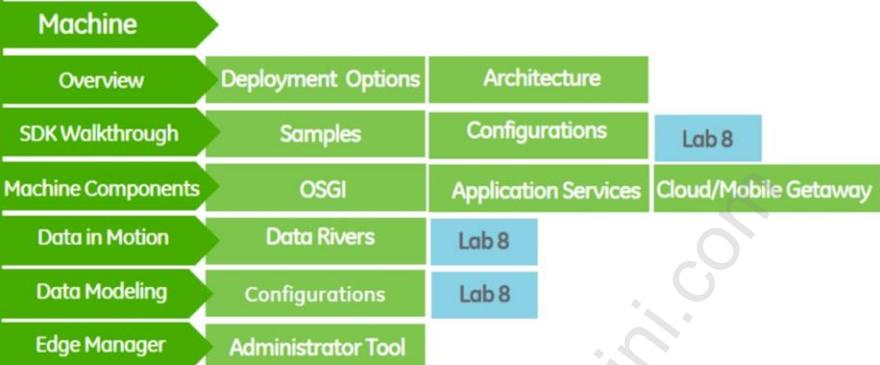


© 2016 General Electric Company - All rights reserved

304



Module Overview: Predix Machine



© 2016 General Electric Company - All rights reserved.

305



Module Objectives: Predix Machine

By the end of this module you will be able to:

- Understand Predix machine architecture
- Generate a working Machine using Predix Software Development Kit (SDK)
- Explore Web console and command line admin options for monitoring and Debugging
- Modify scripts to configure and customize Predix machine
- Move Data from Machine Gateway to Predix cloud using HTTP and WebSocket rivers
- Describe the Machine Data Flow and data models

Completion of the following lab activities will reinforce the concepts covered in this module:

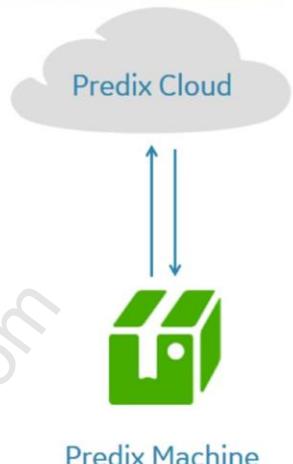
- Lab 8 : Connecting Machines to Predix Cloud



What is Predix Machine

Industrial Internet of Things (IIOT)

- **Predix Machine** is a software solution for making **industrial assets “Internet ready”** and connecting them to the Predix Cloud
- Connects intelligent machines to cloud
- Enables asset management without service interruption
- Provides common runtime for machine applications



© 2016 General Electric Company - All rights reserved

307

- SDM (software defined machine)
- By decoupling the machine software from the hardware, it enables the development, deployment, and management of applications on embedded hardware connected to physical machines
- OSGi framework is another open-source component used in Predix. We use a OSGi version created by Prosys
- It provides a common runtime environment for machine applications allowing industrial operators to deploy, update, maintain, and improve assets without service interruption

By the end of this training, you'll understand the technologies, tools and services provided by Predix machine.



Predix Machine- Use Case



Data Collection

- **Predix Machine** collects data from sensors
- Edge analytics examines and filters the data
- Data pushed to **Predix Cloud**

Predictive Maintenance

- Edge analytics evaluates asset data
- Anomaly detected? => Order shutdown
- New patterns identified
- Problem Severity contained



© 2016 General Electric Company - All rights reserved

308

Let's take a real-life use case to demonstrate how businesses use Predix. Predictive Maintenance is a hot topic these days. Being able to better predict when a problem might occur and proactively shutting down an industrial asset BEFORE the problem occurs, allows businesses to avoid costly, unplanned down times.

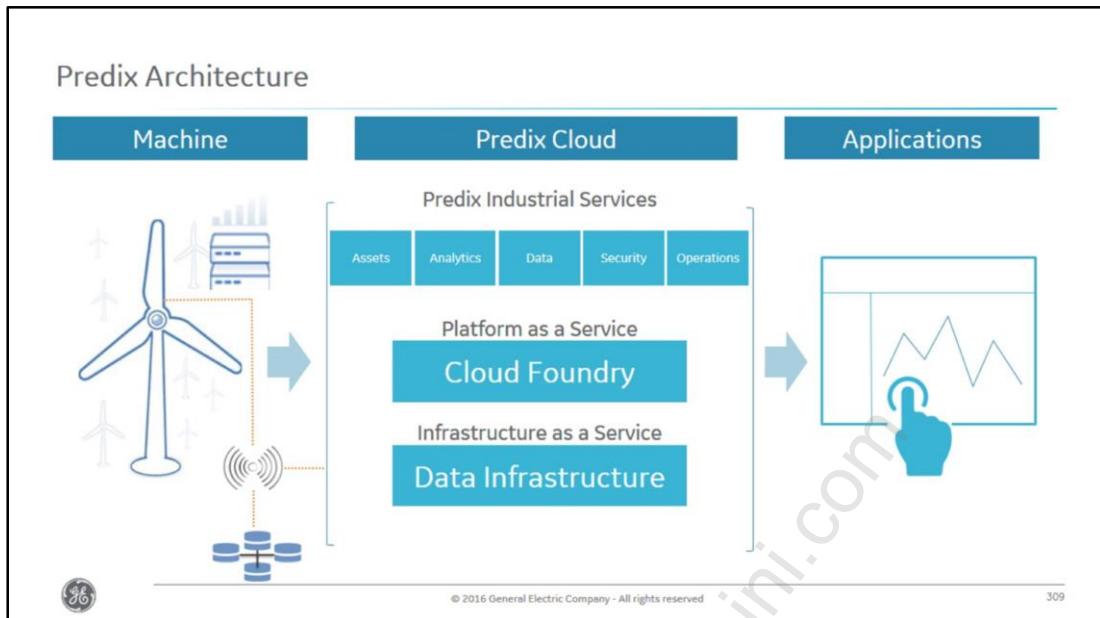
How is this achieved with the Predix platform?

Predix Machine collects data from sensors, in this case a wind turbine, and uses edge analytics to monitor the status of an asset. If something out of the ordinary is detected, Predix Machine can send an alert to the field agent to shut down the turbine gracefully before a serious problem occurs that could damage the turbine, pulling it offline for an extended period of time.

There are still going to be unplanned downtimes that occur for wind turbines. Predix Machine can also pass the operations data from wind turbines to the **Predix Cloud**, where data for all wind turbines can be stored and analyzed. As data is analyzed over time, new patterns may be identified, new edge analytics can be created and fed back to the wind turbines.

The result: decreasing the number of unplanned down times.





The Predix Platform and tools provide a set of software development tools and best practices that enable our customers and partners to optimize their industrial business processes.

Lets take a look at Predix Architecture using the example of predictive maintenance.

Being able to better predict when a problem might occur and proactively shutting down an asset before it causes costly damage or unplanned downtime is just one example of how Predix is being used.

Here we see our industrial asset, the wind turbine. This turbine and the turbine farm sit on the 'edge' of the Predix platform. Using Predix, the data from the wind turbine can be received and analyzed. We can also review and optimize operation to gain maximum value of this asset.

Predix Machine using data collected from the sensors can use edge analytics to monitor the status of an industrial asset. If something out of the ordinary is detected, Predix machine can tell the asset to shut down with control before something catastrophic happens

Predix machine can also pass the operational data for this wind turbine and this windfarm to Predix cloud.

With Predix cloud, all data across all windfarms can be stored and analyzed by data scientists. These data scientists can look for trends over time, identify new patterns, create new edge analytics and push that information back out to ALL wind turbines.

Predix machine is the software stack that can be installed locally on gateways, industrial controllers, or on sensors. Predix Machine is also responsible for executing edge analytics, and communicating with industrial assets as well as the cloud.

Predix connectivity allows Predix machine to talk to the cloud even when 'normal' internet may not be available. This can be common in remote locations like oil rigs.

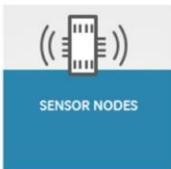
The Predix cloud provides industrial services that developers can use to build test and run industrial internet applications. This is built on a custom built cloud data infrastructure. The data infrastructure is optimized with enhanced security controls and world class data processing and networking capabilities.

Predix Cloud also provides developers an application framework to use industrial microservices as well as a modular design system that allows developers to rapidly deliver innovative applications and a context driven user experience.

From improved analytics, real time asset optimization, or predictive maintenance, the Predix platform is designed to support the continuous improvement of industrial business processes.



Deployment Options



SENSOR NODES

Sensors (temp/flow/
pressure/etc.) and
hubs, I/O devices,
actuators, logic
systems

Nodes | Sensors | Assets –
Endpoint generating the data.
May or may not have network
connectivity.



© 2016 General Electric Company - All rights reserved

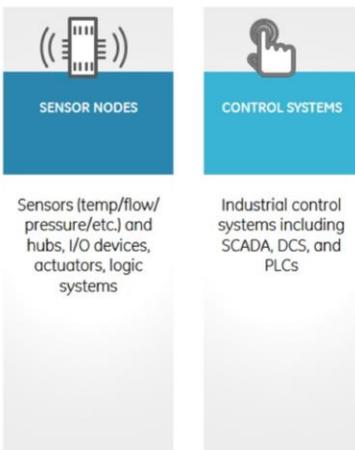
310

Depending on your assets, you can have a few deployment options:

- Devices/sensors built before the internet was available (>1980s) had to physically connect to "something". Many of those are still in play on production floors. The configuration requires a physical connection to a controller that processes data and connects to an internet-ready device
- Sensors today have more options on connectivity -- still lightweight but wifi, bluetooth, zigbee, zwave – so the need for the controller goes away
- You can install on one or more gateways that talk to the cloud. You can have one master gateway that collects data from all the other gateways



Deployment Options



Nodes | Sensors | Assets –
Endpoint generating the data.
May or may not have network
connectivity.

Controller – Hardware and
software for controlling
non-programmable assets;
for example controllers
connected to assets
without network
connectivity.



© 2016 General Electric Company - All rights reserved

311

Depending on your assets, you can have a few deployment options:

- Devices/sensors built before the internet was available (>1980s) had to physically connect to "something". Many of those are still in play on production floors. The configuration requires a physical connection to a controller that processes data and connects to an internet-ready device
- Sensors today have more options on connectivity -- still lightweight but wifi, bluetooth, zigbee, zwave – so the need for the controller goes away
- You can install on one or more gateways that talk to the cloud. You can have one master gateway that collects data from all the other gateways



Deployment Options



[Nodes](#) | [Sensors](#) | [Assets](#) – Endpoint generating the data. May or may not have network connectivity.

Controller – Hardware and software for controlling non-programmable assets; for example controllers connected to assets without network connectivity.

Gateway – Dedicated hardware with connectivity that Predix machine runs on (think router or hub).



© 2016 General Electric Company - All rights reserved

312

Depending on your assets, you can have a few deployment options:

- Devices/sensors built before the internet was available (>1980s) had to physically connect to "something". Many of those are still in play on production floors. The configuration requires a physical connection to a controller that processes data and connects to an internet-ready device
- Sensors today have more options on connectivity -- still lightweight but wifi, bluetooth, zigbee, zwave – so the need for the controller goes away
- You can install on one or more gateways that talk to the cloud. You can have one master gateway that collects data from all the other gateways



Deployment Options



Nodes | Sensors | Assets -
Endpoint generating the data.
May or may not have network connectivity.

Controller – Hardware and software for controlling non-programmable assets; for example controllers connected to assets without network connectivity.

Gateway – Dedicated hardware with connectivity that Predix machine runs on (think router or hub).



Depending on your assets, you can have a few deployment options:

- Devices/sensors built before the internet was available (>1980s) had to physically connect to "something". Many of those are still in play on production floors. The configuration requires a physical connection to a controller that processes data and connects to an internet-ready device
- Sensors today have more options on connectivity -- still lightweight but Wi-Fi, Bluetooth, zigbee, zwave – so the need for the controller goes away
- You can install on one or more gateways that talk to the cloud. You can have one master gateway that collects data from all the other gateways



Sample Configuration



© 2016 General Electric Company - All rights reserved

314

In this sample configuration, we have a piece of hardware – a sensor node - with a radio frequency interface.

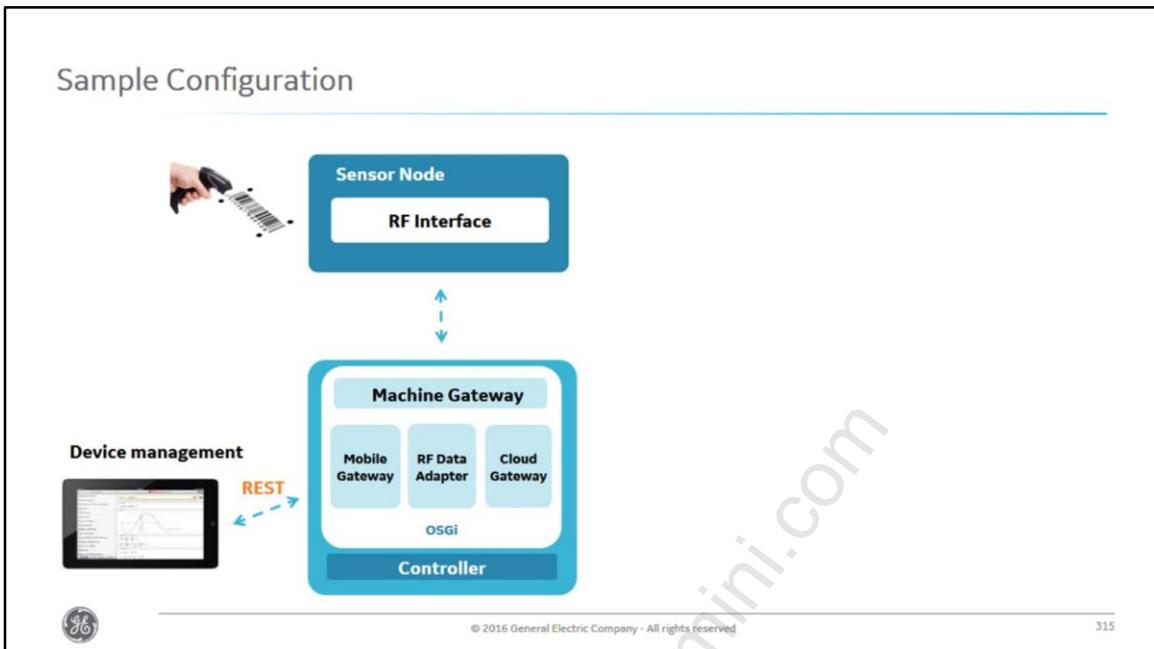
The sensor sends its data to a Gateway with Predix Machine installed on it. It collects the data through a radio frequency adapter installed on the gateway.

Predix Machine (gateway) sends the data to a technician in the field and to the cloud for further analysis and storage.

Based on the results of analytics, Predix machine may possibly send feedback from cloud to the field technician for remote monitoring and control.



Sample Configuration



In this sample configuration, we have a piece of hardware - a sensor node - with a radio frequency interface.

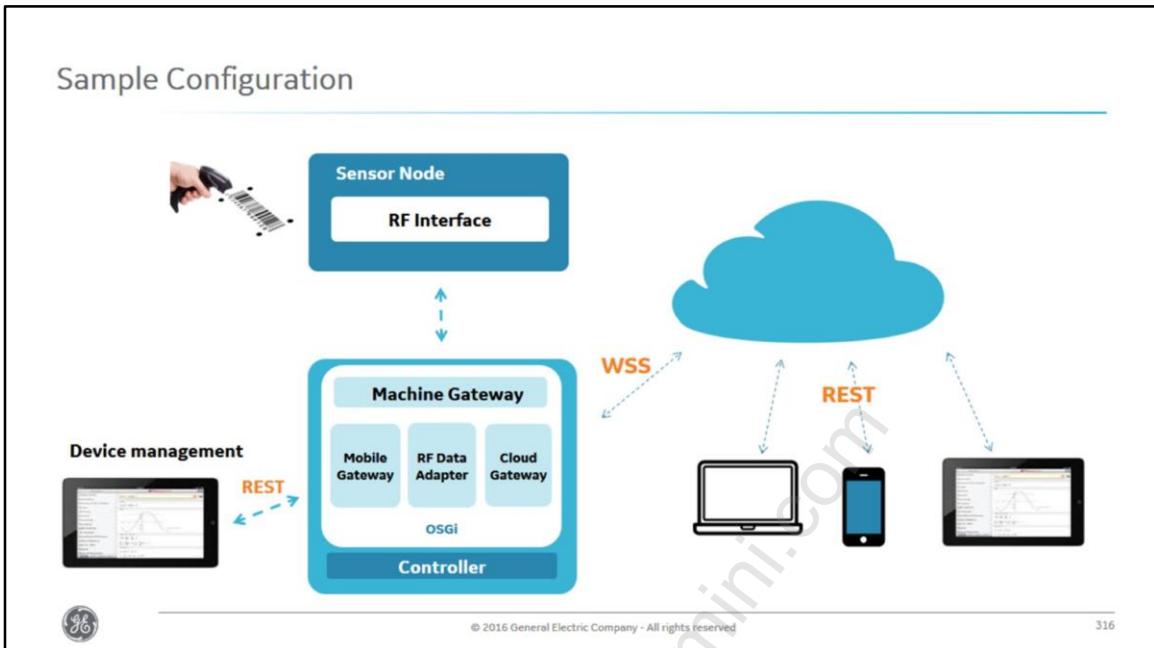
The sensor sends its data to a Gateway with Predix Machine installed on it. It collects the data through a radio frequency adapter installed on the gateway.

Predix Machine (gateway) sends the data to a technician in the field and to the cloud for further analysis and storage.

Based on the results of analytics, Predix machine may possibly send feedback from cloud to the field technician for remote monitoring and control.



Sample Configuration



In this sample configuration, we have a piece of hardware – a sensor node - with a radio frequency interface.

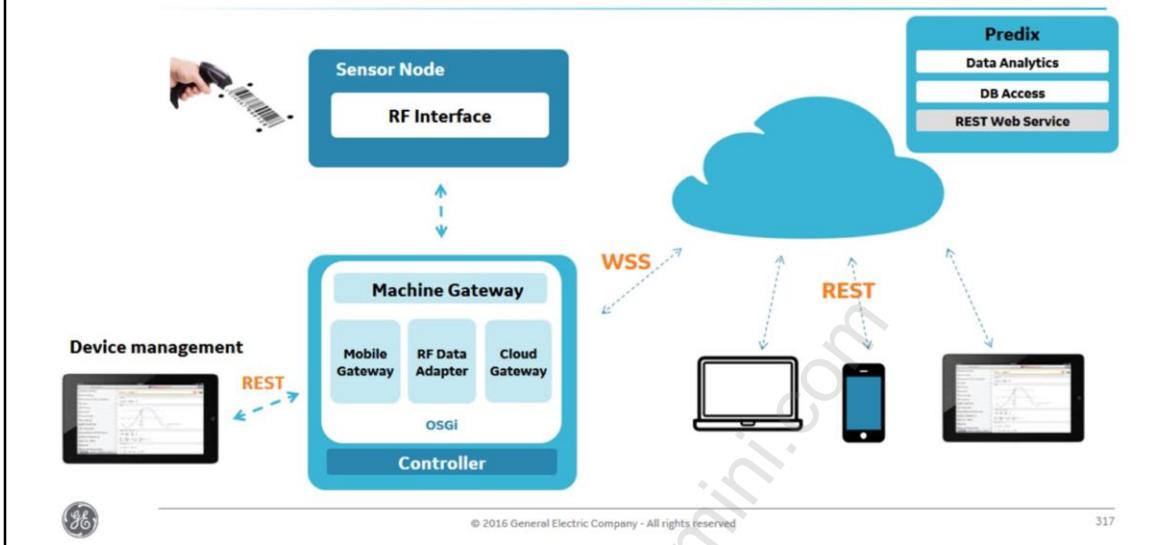
The sensor sends its data to a Gateway with Predix Machine installed on it. It collects the data through a radio frequency adapter installed on the gateway.

Predix Machine (gateway) sends the data to a technician in the field and to the cloud for further analysis and storage.

Based on the results of analytics, Predix machine may possibly send feedback from cloud to the field technician for remote monitoring and control.



Sample Configuration



In this sample configuration, we have a piece of hardware - a sensor node - with a radio frequency interface.

The sensor sends its data to a Gateway with Predix Machine installed on it. It collects the data through a radio frequency adapter installed on the gateway.

Predix Machine (gateway) sends the data to a technician in the field and to the cloud for further analysis and storage.

Based on the results of analytics, Predix machine may possibly send feedback from cloud to the field technician for remote monitoring and control.



Predix Machine

Machine Architecture

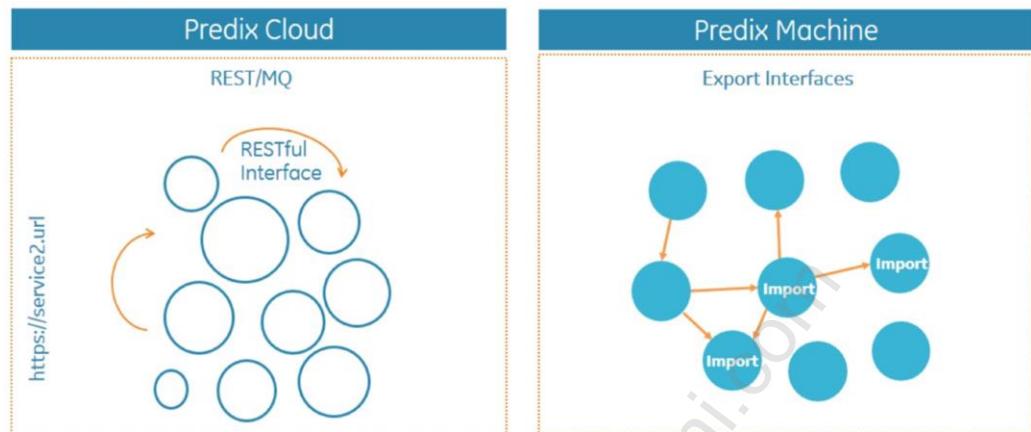


© 2016 General Electric Company - All rights reserved

318



Predix Services

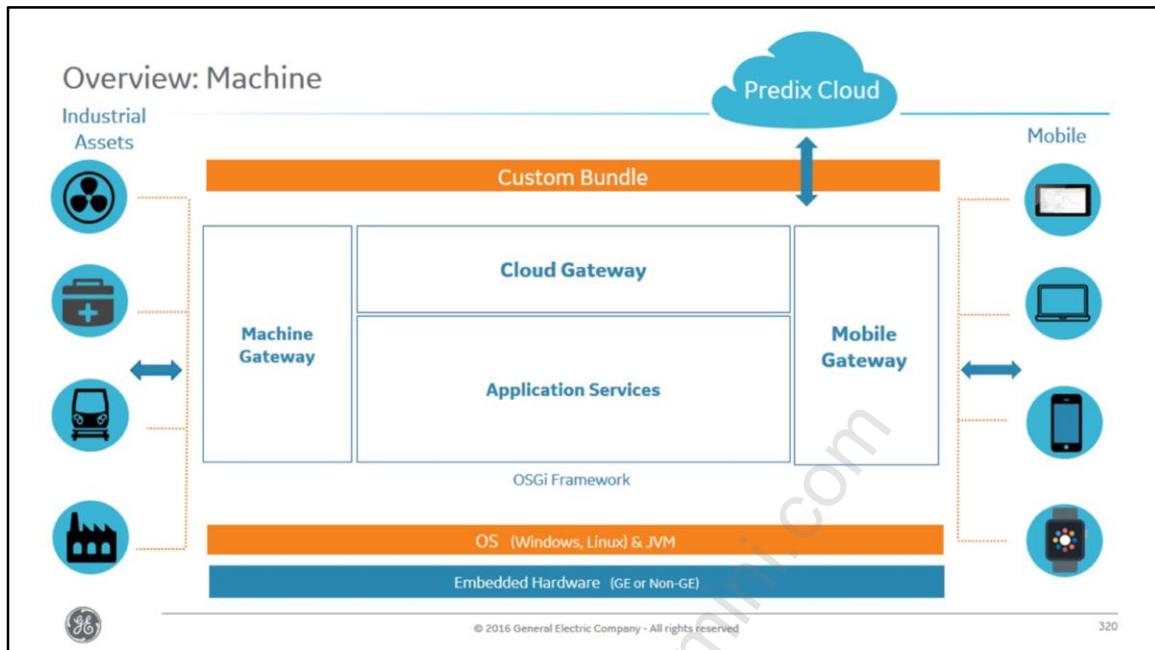


© 2016 General Electric Company - All rights reserved

319

- SOA in Cloud vs. Machine
- So Predix cloud employs microservices, which communicated with each other through RESTful APIs or MQ services
- In comparison, Predix machine is a service –oriented bundle in OSGi framework where individual services import interface from another service (e.g. UAA) to “consume” it
- Cloud microservices are language agnostic, however Machine bundles are developed in Java only
- No REST interfaces – SOA bot not microservices





Architecturally, Machine has a very small footprint. It's a combination of various technologies that support different features.

- At the bottom you have your host platform (operating System) – windows or Linux, and JVM
- The OSGi framework is a “Prosyst” implementation of OSGi (vs Apache Karaf) and comes with a host of services to complement Machine utilities. You can have multiple OSGI containers running in one JVM
- To the left is the Machine Gateway. It provides M2M services to communicate between gateways and/or sensors
- In the middle, there are “Application Services” that provide support for the applications you create (device provisioning, data management, edge analytics, etc.)
- Above the “Application Services” is the Cloud Gateway that services your connection between Machine and Predix Cloud
- To the Right, the “Mobile Gateway” provides services that connect your mobile devices to Machine
- Create custom bundles to provide the additional Px Machine functionality that you require



Predix Machine

Predix SDK Walkthrough - Generate a Machine



© 2016 General Electric Company - All rights reserved

321



Where to Get the Predix Machine SDK

Predix.io Catalog > Edge Software and Services



© 2016 General Electric Company - All rights reserved

322

Download the SDK using link:

<https://artifactory.predix.io/artifactory/PREDIX-EXT/predix-machine-package/predixsdk/16.2.0/predixsdk-16.2.0.zip>

Enter the cloud foundry credentials when prompted.

Install the Machine documentation and samples by unzipping the [predixsdk-16.2.0.zip](#) file from the command line in a terminal.



Examine and Import the SDK

/docs

- Javadoc APIs for Predix machine services
- Prosyst docs for OSGi containers

/eclipse-plugins

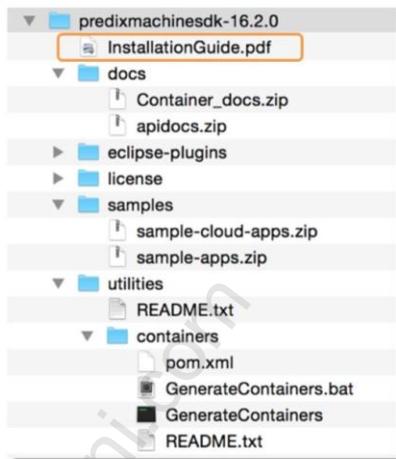
- Import into your Eclipse-based tool (STS)

/samples

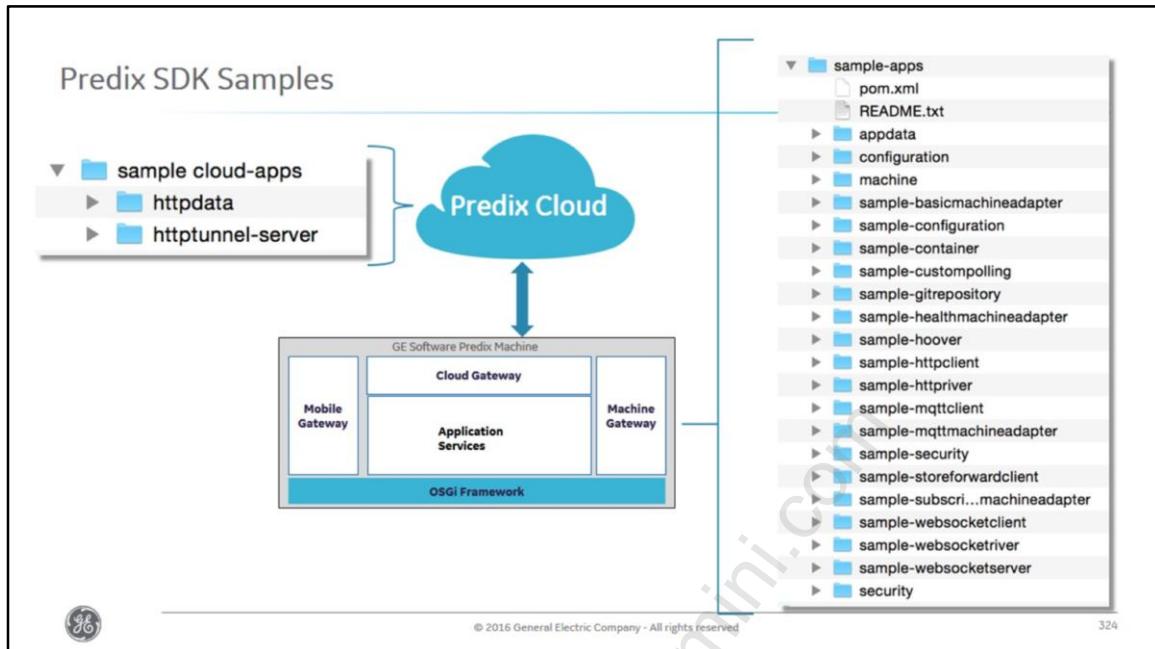
- Predix machine samples
- Predix cloud sample for HTTP data service

/utilities

- Machine containers and scripts



On the Devbox, open the **PredixApps/training_labs/machine/predixmachinesdk-16.2.0** folder and examine the files available.



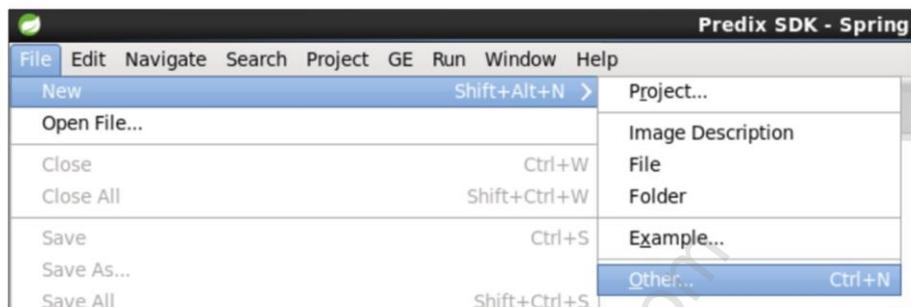
There are 2 sets of samples in the SDK. The first set are samples that run in the predix machine container and use the APIs across the different machine services.

The second set of samples run in the Predix Cloud. One of the key samples is the httpdata service which receives data sent from Predix Machine Data River and writes it to a Postgres database.



Generate a Machine Runtime Container

Step 1: Create a parent project



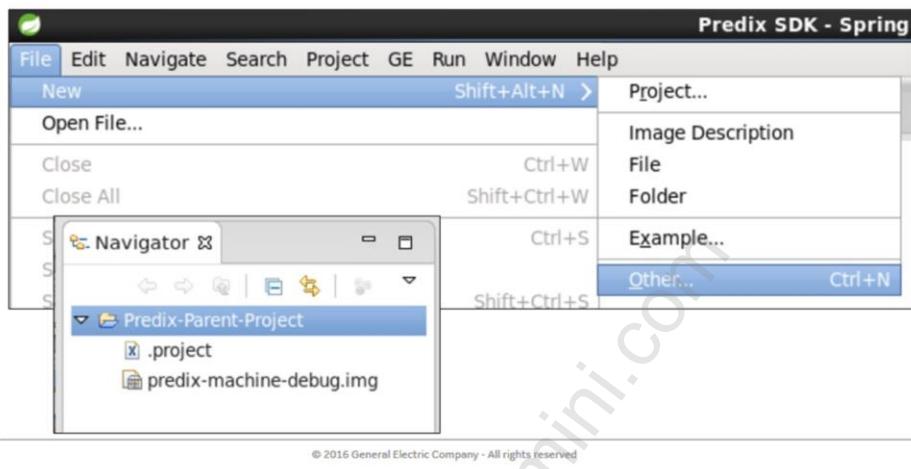
© 2016 General Electric Company - All rights reserved.

325



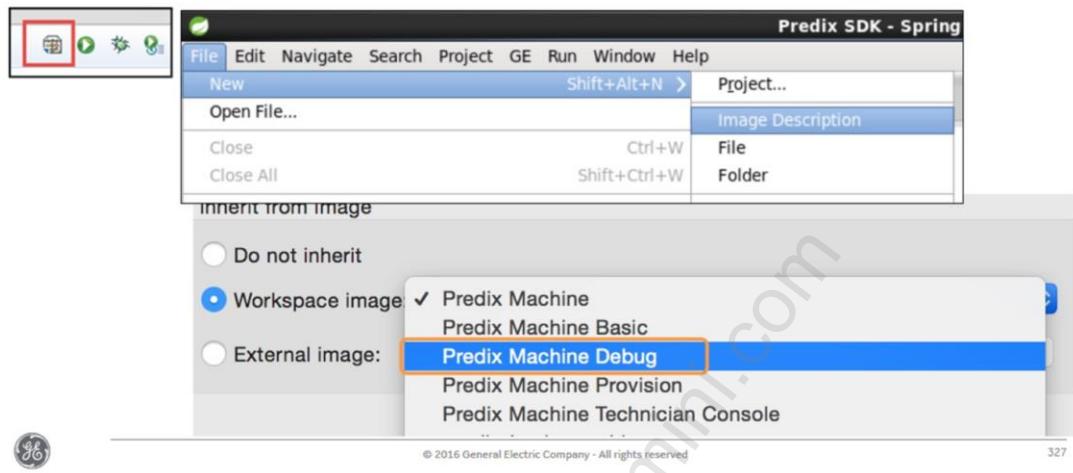
Generate a Machine Runtime Container

Step 1: Create a parent project



Generate a Machine Runtime Container

Step 2: Create a machine image and export



If you prefer to not use Eclipse, command line interface can be used for generating a machine container.

Predix machine is typically generated from within Eclipse in a 2 step process. You create an empty parent project first, then create a new "Image Description". This Image Description is a blueprint of the machine modules, or bundles as they are called, that you selected to be included in your machine container.

Eclipse-generated:

- Create an image (clone) of the Predix machine debug container
- Export the machine container image you created to your project directory
- You will be working with this clone container throughout this lab



Predix Machine Configurations: “Features”

Predix Machine Debug

- 116 overall | 39 machine | 77 runtime
- Used for testing



Predix Machine Provision

- 67 overall | 15 machine | 52 runtime
- Used for automated device enrollment or provisioning
- Also provided in a Docker container on Predix.io as a direct install

Predix Technician Console

- 70 overall | 13 machine | 57 runtime
- Used for manual device enrollment & monitoring of machine by field service engineer



© 2016 General Electric Company - All rights reserved.

328

These are a few of the blueprints available to you in machine.

Predix Machine Debug is a full version of machine that includes everything. It's great for testing but you would never deploy this configuration in the field.

The Provision version of machine contains only the modules needed to provision, or enroll, the hardware with the Edge Manager. The Edge Manager is an administrative tool that allows an operator to monitor all instances of machine on the plant floor.

The Technician Console is another administrative version of machine that contains only the modules needed by a field service engineer on the plant floor that needs to access an individual instance of machine.



Predix Machine Configurations: “Features”

Predix Machine Custom

- Create for a specific purpose
- Moving data most common use case
 - Cloud gateway components
 - Spillway components
 - Data store and forward components
 - Specific river components
 - Optional Web Console component



© 2016 General Electric Company - All rights reserved

329

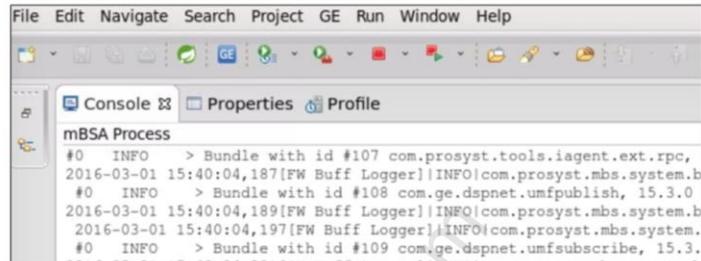
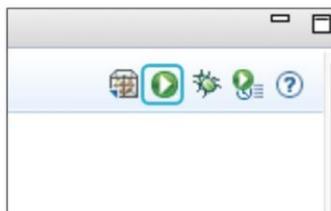
The most common use case for Predix Machine is to collect and move data through a specific protocol. You would configure this by way of feature sets. The feature sets include all the necessary components, or bundles, needed for that task.

You would typically include the Cloud gateway feature, the Spillway feature, Data Store and Forward components, the specific river you're sending the data over, and optionally an administrators web console.



Start the Machine Container

Eclipse Run option



© 2016 General Electric Company - All rights reserved.

330

There are two options for starting your machine container:

1. Starting the container from Eclipse:

- Start the machine from Eclipse by clicking the “run” graphic button
- Open the admin console.
- Select the IMG file and run from the arrow button in the Overview tools bar. The console window shows the startup logs.

2. Starting the container from command line:

- You can start the container at the command line with command: “./start_container.sh clean”.
- Upon successful startup, you’ll see a message: “OSGi framework/1 started successfully”



Start the Machine Container

Command line option: \$./start_container.sh clean

```
[x predix@localhost:~/PredixApps/training_labs/machine/predixmachinesdk-16.2.0/utilities/containers/predix-mac ~]
File Edit View Search Terminal Help
2016-08-26 12:21:34,125[FW Buff Logger] |INFO|com.prosyst.mbs.system.bundle[23
-com.prosyst.mbs.system.bundle-1.0.0|[PlatformState] State Change from STARTI
NG to ACTIVE
Server is started.
```



There are two options for starting your machine container:

1. Starting the container from Eclipse:

- Start the machine from Eclipse by clicking the “run” graphic button
- Open the admin console.
- Select the IMG file and run from the arrow button in the Overview tools bar. The console window shows the startup logs.

2. Starting the container from command line:

- You can start the container at the command line with command: “./start_container.sh clean”.
- Upon successful startup, you’ll see a message: “OSGi framework/1 started successfully”



Machine Utilities

Command line admin options

```
# predix@i01-DevBox:~/PredixApps/training_labs/machine/predixdk-15.3.0/utilities/containers/predix-debug-n ~
File Edit View Search Terminal Help
# 
#  INFO > Bundle with id #112 com.ge.demicro.machinedapter-healthmonitor, 15.3.0 was started.
2014-05-20 12:20:48,450 INFO [INFO]com.geosys.net.bundle[12-com_geosysnet_bundle-1.0.0]
Bundle with id #112
2014-05-20 12:20:48,472 INFO [INFO]com.geosys.net.bundle[12-com_geosysnet_bundle-1.0.0]
Component Remote Thread (bundle 34) [INFO]com.ge.demicro.machinelogateway.impl.MachineLog
ewayImpl[12-com_geosysnetbundleimpl-15.3.0]Machine adapter "[12:0d21e-d011-40ca-940f-3e31251aa10" sta
tus
2014-05-20 12:20:48,484 INFO [INFO]com.geosys.net.bundle[12-com_geosysnet_bundle-1.0.0]
Bundle with id #113 com.ge.demicro.healthmonitorimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #113 com.ge.demicro.healthmonitorimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #114 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #115 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #116 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #117 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #118 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #119 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #120 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #121 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #122 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #123 com.geosys.net.gatewayimpl, 15.3.0 was started.
# 
#  INFO > Bundle with id #124 com.geosys.net.gatewayimpl, 15.3.0 was started.
```

Web Console admin options
(localhost:8443/system/console/)



Predix Machine Web Console Log Service

Main OSGI Predix Status System Technician Console Web Console Log out

Log Service is running.

Exception details: Message Only <input checked="" type="checkbox"/> Severity at least: DEBUG <input type="checkbox"/> Reload						
Received	Level	Message	Service	Bundle	Exception	
8/26/2016, 3:11:09 PM	INFO	User "predix" successfully logged in.		Predix Machine Management Impl		
8/26/2016, 3:10:57 PM	INFO	Password has been created/changed for user "predix"		Predix Machine Management Impl		
8/26/2016, 3:10:57 PM	INFO	User "predix" successfully logged in.		Predix Machine Management Impl		

© 2016 General Electric Company - All rights reserved.

332

For testing purposes, you want to create “the Debug” container and use the Admin Console.

The Admin console lets you start and start any bundles; including ones you create.

After you create your own bundles, you can add them to the container through an install/update button.

You can also examine details on any bundle (exported or imported packages, directory for bundle, manifest properties)

Open a web browser and navigate to: “https://localhost:8443/system/console”

The web console will open upon accepting exceptions and entering your login credentials.



Machine



Lab 8: Connecting Machines to Predix Cloud

Part I: Generate a Debug Machine Container

- Exercise 1: Generate a debug machine container
- Exercise 2: Test container management options



© 2016 General Electric Company - All rights reserved.

333

Exercise 1: Generate a Debug Machine Container

The Predix SDK contains all the scripts and documentation to generate an OSGI container as well as bundles that run inside the container.

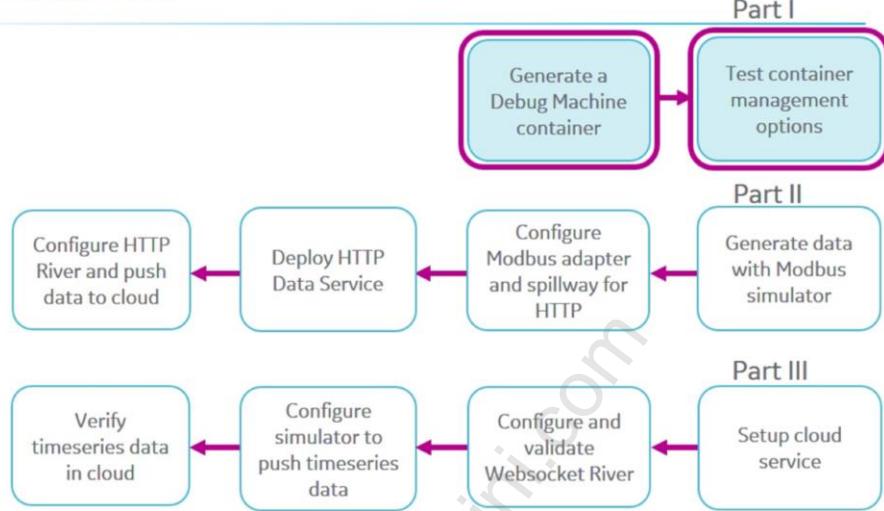
Exercise 2: Test container management options

The Debug container can be managed in one of two ways: in a terminal window through command line options, or through a web interface.

In this exercise, you will explore the two options.



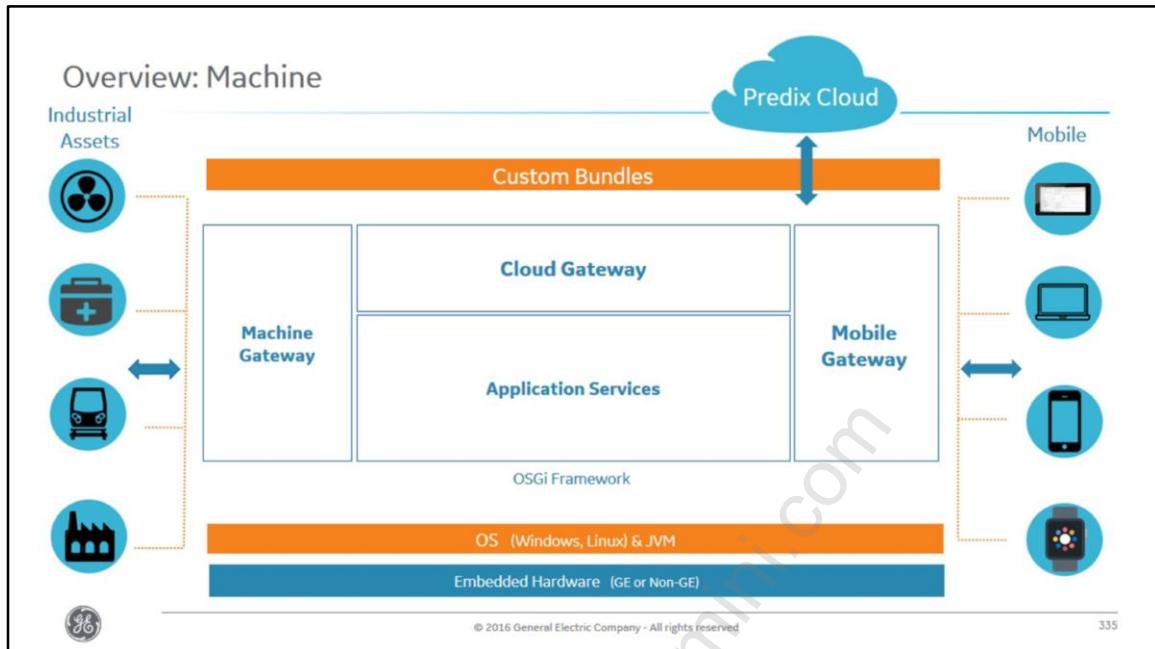
Machine Lab Process Flow



© 2016 General Electric Company - All rights reserved

334





Now that you've had a chance to generate a working version of Predix machine, let's go back and look at the architecture and feature sets available in machine.

Architecturally, Machine has a very small footprint. It's a combination of various technologies that support different features.

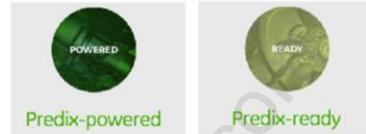


Hardware | Software

Hardware Agnostic



GE Digital Alliance Program
www.predix.com/partners



OS (Windows, Linux) & JVM

Embedded Hardware (GE or Non-GE)

© 2016 General Electric Company - All rights reserved.

336

If we start at the bottom, we're looking at the hardware that machine can run on. Predix machine is hardware agnostic, meaning it can run on any hardware configuration that supports either linux or windows. We don't care whether it's using an ARM chipset or X86. In this slide we show the array of reference boards that machine has successfully been tested on.

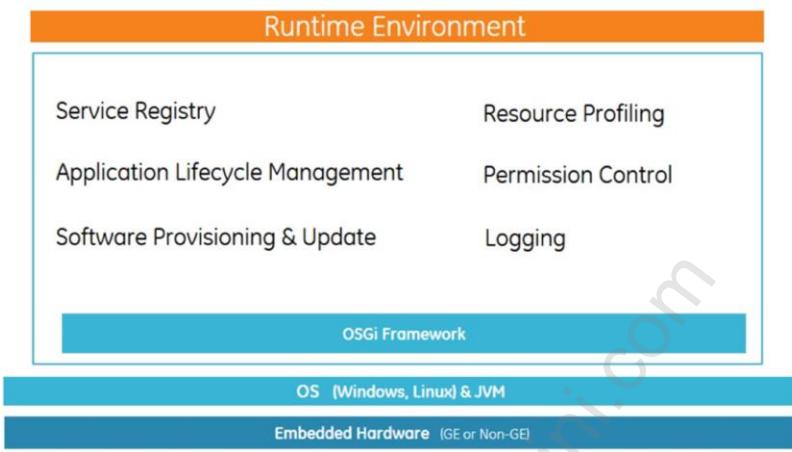
To expedite the deployment of machine in the field, GE is working on a program called Predix-Ready that brings 3rd party vendors together to create production-ready hardware with preloaded versions of Predix machine.

- Predix-powered
 - Independent software vendors (ISVs) interested in app development, managed services, global cloud deployment, and opportunities to leverage the Predix marketplace.
- Predix-ready
 - Technology partners (OEMs and ODMs) interested in embedded Predix Machine software/product integration, software-as-a-service (SaaS), and managed services.

GE Digital Alliance program



OSGi Framework



- OSGI is your runtime environment.
- Everything in OSGI (including applications) is called a "BUNDLE".
- Every installed bundle is registered with the Service Registry. It manages the bundle interactions.
- The runtime also keeps track of bundle lifecycles – start, stop, installed, resolved, etc.
- You can update any individual bundle without having to shut down other bundles with the software provisioning and update service
- There is resource profiling, permission control, and logging all provided by the runtime.



The diagram illustrates the layered architecture of Application Services. At the top is a yellow box labeled "Application Services". Below it is a blue box labeled "OSGI Framework". Further down is a teal box labeled "OS (Windows, Linux) & JVM". At the bottom is a dark blue box labeled "Embedded Hardware (GE or Non-GE)". A watermark "predixmechanical.com" is visible across the diagram.

Certificate Management: Keystores Truststores – secures your data through public/private keys.	Data Transfer: Data Store and Forward – stores data when connectivity is lost and forward when connection is live.	Pluggable Data Processor: Edge analytics – custom module that plugs in and runs minimal analysis on data.
--	--	---

© 2016 General Electric Company - All rights reserved. 338

Now we get to the machine-specific components or feature sets. The first is Application Services.

Every piece of equipment running machine must have a valid CA (certificate authority) generated certificate in order to connect to Predix Cloud. The Certificate management service manages the keystore for your certificate. This is one of 2 ways Predix Machine secures your environment.

If you lose connectivity to Predix Cloud, machine has a component, called Data store and Forward, that stores the data for you until connectivity is returned.

If you want to clean up the data before it's pushed to the cloud, you can create an edge analytic that plugs into machine. Keep in mind that edge analytics are intended for lightweight filtering of the data, not analytic trends or reporting.

NOTE: Postgres service allows any type of data, Time series only supports numeric data (no strings).

Some sensors/devices or mobile applications may be disconnected periodically or even frequently though continuous data is desired.

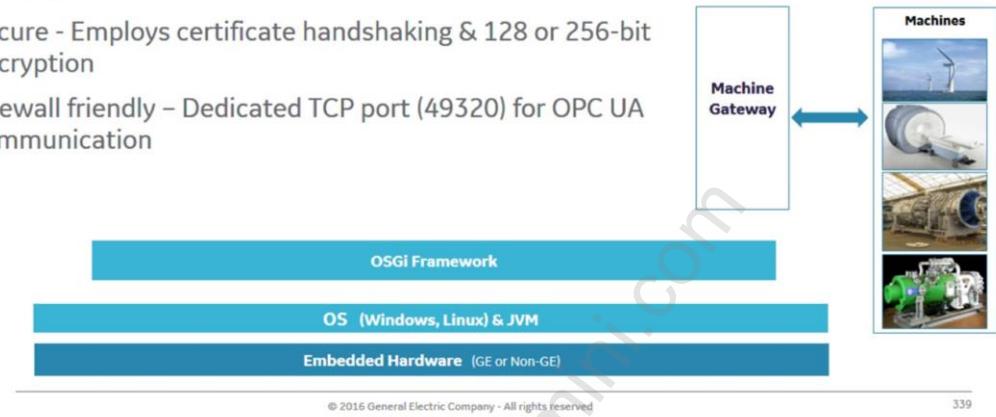
Predix Machine applications may cache data locally when connections are down.



Machine Gateway

OPC-UA (Open Platform – Uniform Architecture)

- Protocol moves data between software systems, devices & sensors
- Secure - Employs certificate handshaking & 128 or 256-bit encryption
- Firewall friendly - Dedicated TCP port (49320) for OPC UA communication



The next feature set is the Machine Gateway.

The machine gateway manages the connection between your assets, and the Predix machine platform. It's the "gateway" for your data. Your assets, or sensors, will be configured to communicate over a specific protocol. In the Machine Gateway we provide modules that support three industry protocols – OPC-UA, Modbus, and MQTT.

OPC-UA is the most secure of the protocols. It provides 128 or 256-bit encryption and is fire-wall friendly. It is also the heaviest of the three protocols.

OPC-UA: TRANSPORT

- A transport is the mechanism that moves an OPC UA message between a client and server. All OPC UA messages are delivered over a TCP/IP connection.
- OPC UA only uses dedicated TCP port (49320) for communication, so it is considered firewall friendly.



Machine Gateway

OPC-UA (Open Platform – Uniform Architecture)

Modbus

- Message-centric protocol for quick, reliable communication of data
- Focus on message delivery, regardless of data payload
- Straightforward connections to devices
- Offers no security, such as data encryption



© 2016 General Electric Company - All rights reserved.

340

Modbus

- Message-centric protocol ideal for quick, reliable communication of simple data to and from I/O devices, while consuming low bandwidth (very lightweight)
- Focus is on delivery of the message itself regardless of the data payload it contains. The infrastructure's role is to ensure that messages get to their intended recipients. Typically used for straightforward connections to devices
- Not for complex device modeling or for connecting on a software systems level. Offers no security, such as data encryption



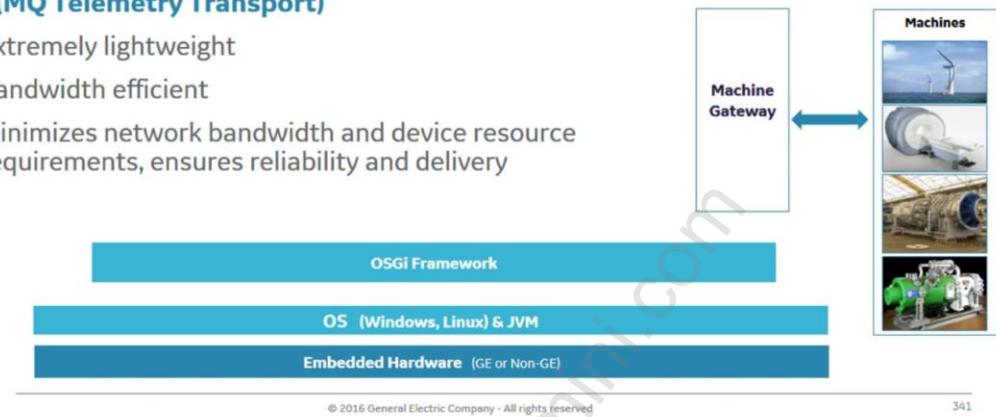
Machine Gateway

OPC-UA (Open Platform – Uniform Architecture)

Modbus

MQTT (MQ Telemetry Transport)

- Extremely lightweight
- Bandwidth efficient
- Minimizes network bandwidth and device resource requirements, ensures reliability and delivery



341

The third protocol supported is MQTT (MQ Telemetry Transport). This is a very light-weight protocol used when resources, like memory, network, or disk space, are constrained. It uses a publish/subscribe messaging transport ideal for these kinds of devices.

MQTT (MQ Telemetry Transport)

- Extremely lightweight client-server publish/subscribe messaging transport
- Designed for resource constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimize network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery
- Ideal protocol for the emerging “machine-to-machine” (M2M) or “Internet of Things”(IoT) world of connected devices, and for mobile applications where bandwidth and battery power are at a premium
- Standard ports for MQTT
 - TCP/IP port 1883 is reserved with IANA for use with MQTT
 - TCP/IP port 8883 is also registered, for using MQTT over SSL
- Supports security
 - You can pass a user name and password with an MQTT packet in V3.1 of the protocol. Encryption across the network can be handled with SSL, independently of the MQTT protocol itself (it is worth noting that SSL is not the lightest of protocols, and does add significant network overhead)
 - Additional security can be added by an application encrypting data that it sends and receives, but this is not something built-in to the protocol, in order to keep it simple and lightweight



Machine Gateway

OPC-UA (Open Platform – Uniform Architecture)

Modbus

MQTT (MQ Telemetry Transport)

APIs for Custom Solution

- Machine Gateway APIs creates custom solution (adapter)

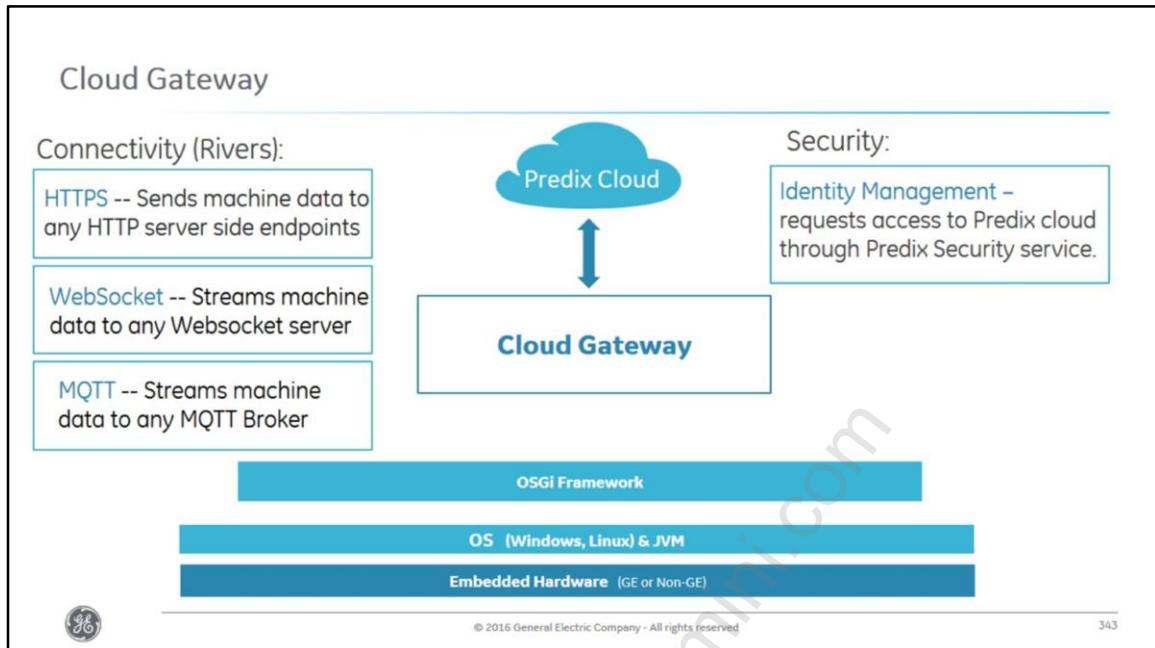


© 2016 General Electric Company - All rights reserved

342

In addition to the three protocols discussed so far, we provide APIs to create custom adapters for protocols such as Zigbee or Zwave. This allows to collect data coming in from sensors over a different protocol.



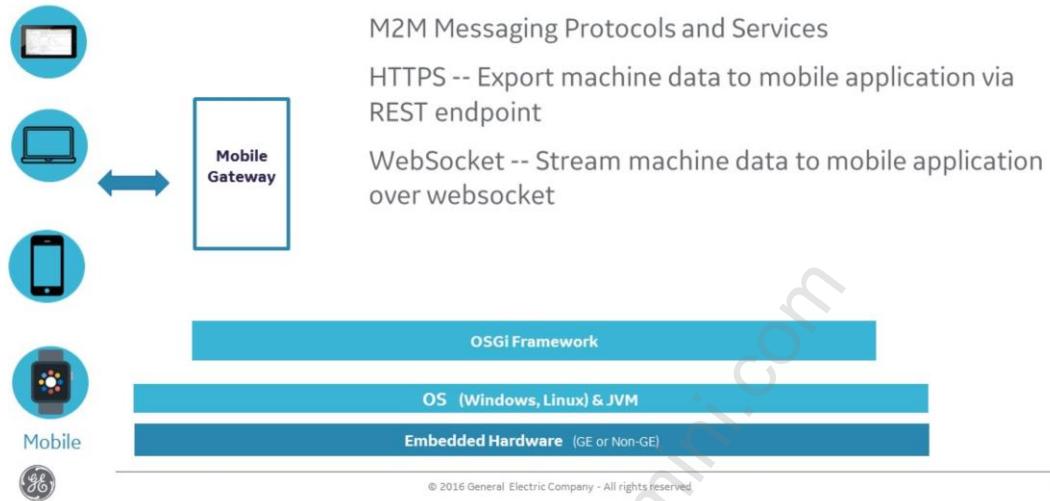


Cloud Gateway

- Set of components that make up Cloud gateway in machine, made up of bundles
- Pushes data to the cloud through data rivers – HTTPS, Web Socket (DDS, i.e. data distribution service) rivers Security/Identity Management
 - Registers Predix machines with cloud services.
 - Provides client credential validation through the security service (UAA).
- Websocket is the more widely used protocol
- It leaves the connection open, so is a much quicker solution
- HTTPS is not the fastest choice as closes the connection between sends.



Mobile Gateway



Mobile Gateway provides services that connect your mobile devices to Machine:

- No separate components support mobile gateway
- Mobile gateway can display admin console on mobile device
- IP address and hardware running machine
- Mobile- can write own app (bundle) to access machine info and bring it to a mobile device (table).



Predix Machine

Data in Motion: Moving Data from Machine Gateway to the Data River



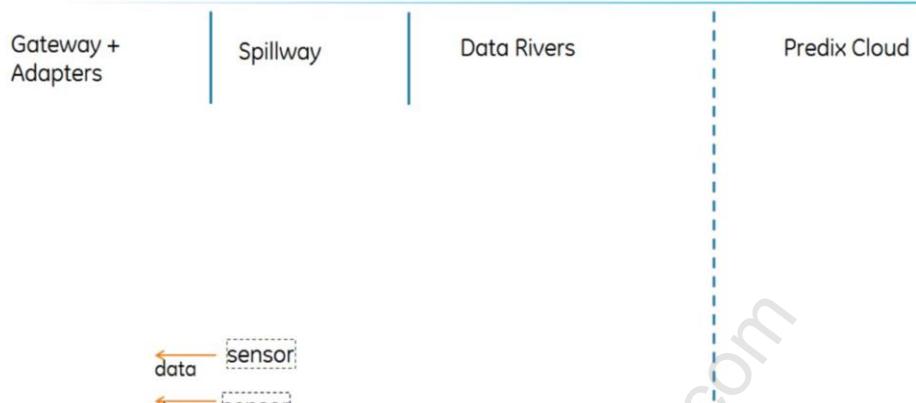
© 2016 General Electric Company - All rights reserved

345

Now that you have an introductory perspective on the feature sets in machine, let's look at how those features are used in a typical configuration. The most common configuration for machine is to move data from the Machine Gateway to one of the Data Rivers. We call this "Data in Motion".



Data in Motion: Overview



© 2016 General Electric Company - All rights reserved

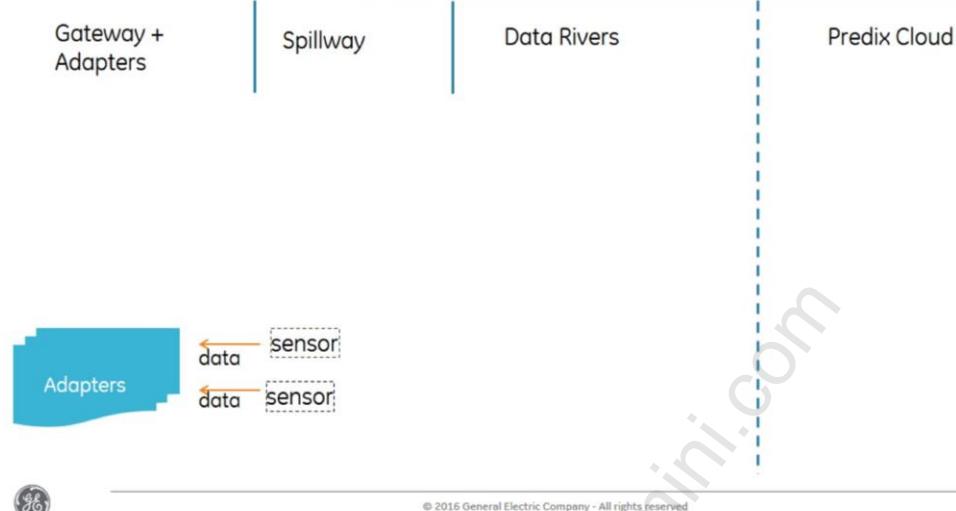
346

When we look at data in motion, we have to start with the sensors. The sensors generate data, which is read by an adapter. The adapter sends the data through the machine gateway to the hoover spillway. The spillway is configured to send the data to a specific river. Depending on the river, your data will be pushed to a data management service in Predix Cloud. That's the high level picture of data in motion.

Objective is to move the data generated by asset sensors to the Predix cloud. Adapters and Machine Gateway bring the data into data rivers, which transport the data to the cloud for storage and analytics.



Data in Motion: Overview

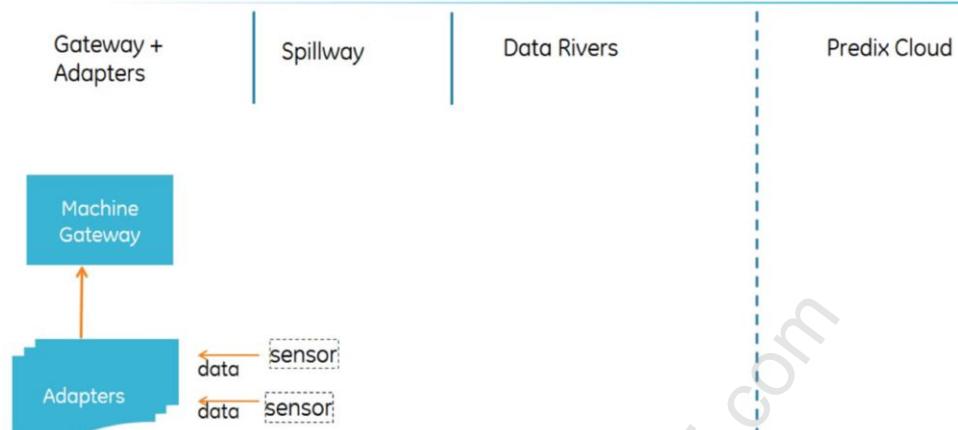


When we look at data in motion, we have to start with the sensors. The sensors generate data, which is read by an adapter. The adapter sends the data through the machine gateway to the hoover spillway. The spillway is configured to send the data to a specific river. Depending on the river, your data will be pushed to a data management service in Predix Cloud. That's the high level picture of data in motion.

Objective is to move the data generated by asset sensors to the Predix cloud.
Adapters and Machine Gateway bring the data into data rivers, which transport the data to the cloud for storage and analytics.



Data in Motion: Overview



© 2016 General Electric Company - All rights reserved.

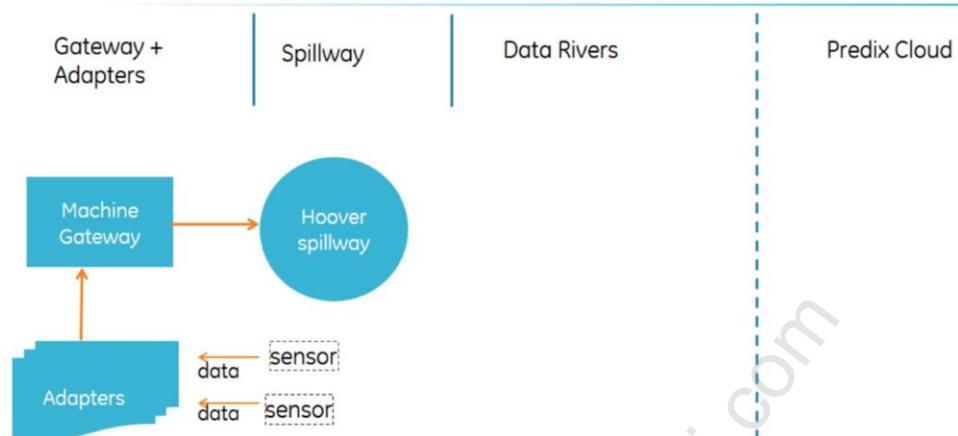
348

When we look at data in motion, we have to start with the sensors. The sensors generate data, which is read by an adapter. The adapter sends the data through the machine gateway to the hoover spillway. The spillway is configured to send the data to a specific river. Depending on the river, your data will be pushed to a data management service in Predix Cloud. That's the high level picture of data in motion.

Objective is to move the data generated by asset sensors to the Predix cloud. Adapters and Machine Gateway bring the data into data rivers, which transport the data to the cloud for storage and analytics.



Data in Motion: Overview

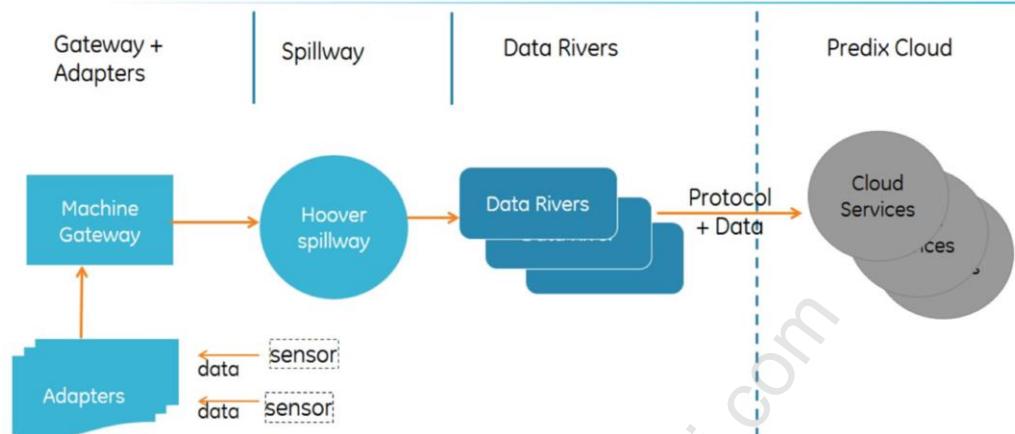


When we look at data in motion, we have to start with the sensors. The sensors generate data, which is read by an adapter. The adapter sends the data through the machine gateway to the hoover spillway. The spillway is configured to send the data to a specific river. Depending on the river, your data will be pushed to a data management service in Predix Cloud. That's the high level picture of data in motion.

Objective is to move the data generated by asset sensors to the Predix cloud. Adapters and Machine Gateway bring the data into data rivers, which transport the data to the cloud for storage and analytics.



Data in Motion: Overview



© 2016 General Electric Company - All rights reserved.

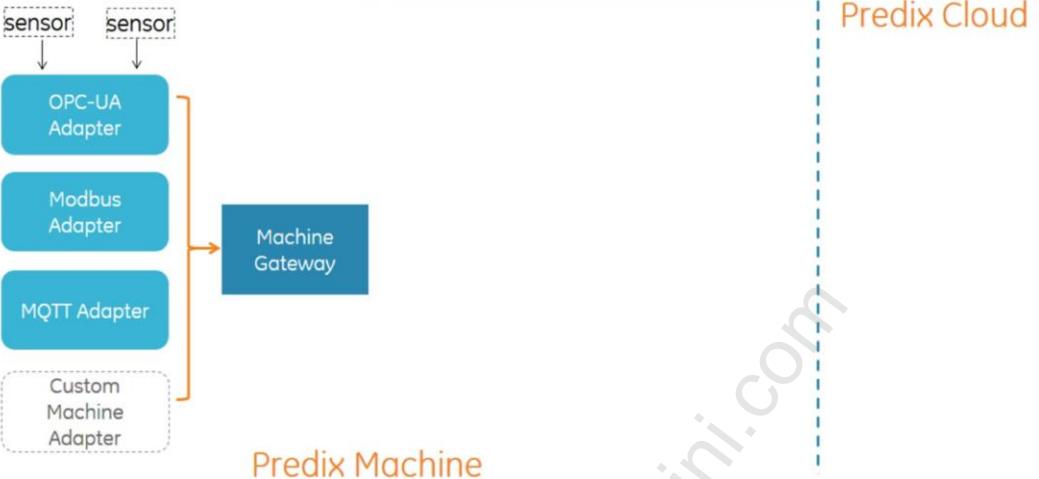
350

When we look at data in motion, we have to start with the sensors. The sensors generate data, which is read by an adapter. The adapter sends the data through the machine gateway to the hoover spillway. The spillway is configured to send the data to a specific river. Depending on the river, your data will be pushed to a data management service in Predix Cloud. That's the high level picture of data in motion.

Objective is to move the data generated by asset sensors to the Predix cloud.
Adapters and Machine Gateway bring the data into data rivers, which transport the data to the cloud for storage and analytics.



Machine Data Flow – Data in Motion



This is what you get. Starting on the left, the machine adapters include OPC-UA, Modbus, MQTT adapter. You can also create your own adapter if these don't service your need.

Sensor data comes into the adapter and is pushed to the Hover Spillway through the machine gateway.

The spillway is the traffic cop that is configured to send data to a specific River. If you need to run analytics on your data, you can write your own "Edge Processor" module that plugs into the spillway. It's an optional component but will act on your data before it goes to any of the rivers.

Currently, we support 3 types of Rivers: HTTP, WebSocket, and MQTT. The data going through all 3 is batched as a JSON objects going over a secure connection.

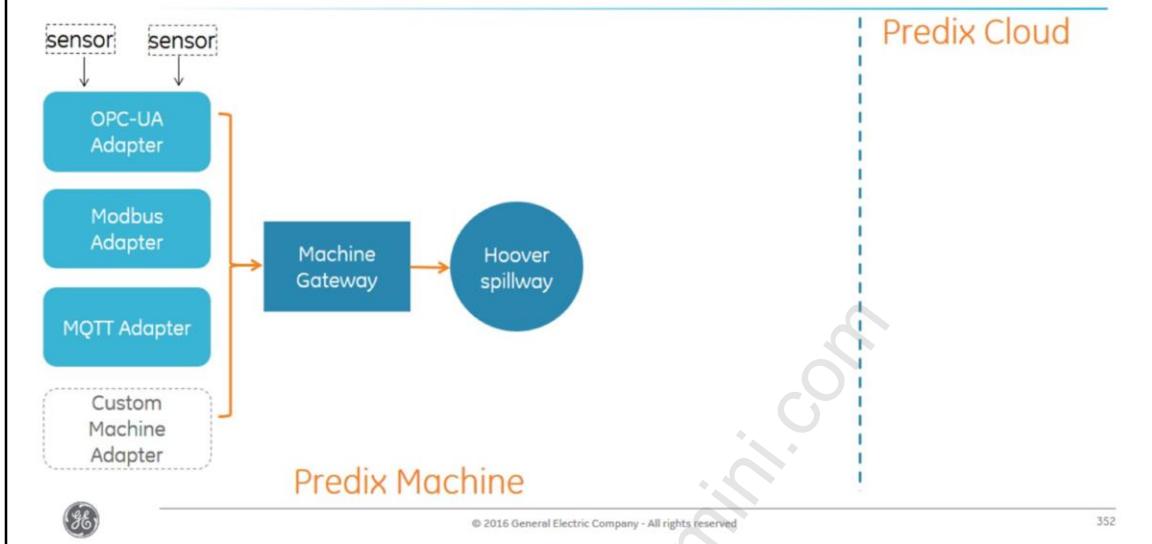
The HTTP river sends the data through HTTPS to an HTTP endpoint in the cloud. The data can be stored in any data management service that makes sense for your use case. The nature of HTTP is to open and close the connection with every request so you may see a measured interval between batches.

The WebSocket river sends data over a websocket. Because websockets stay open until you close them, the pipeline for data going to the cloud is much larger than HTTP. The built-in WebSocket River is designed to connect directly to your timeseries instance in the cloud, therefore you must use the Time Series service to store your data. If you need to use a different data store, you can create your own WebSocket River and configure the spillway to point to it.

If you chose to use the MQTT River, your data will be sent to an MQTT Broker configured to accept the incoming connection request. The Broker can be sitting anywhere, including another installation of Predix Machine. This is how you would daisy-chain your instances of machine together on a plant floor.



Machine Data Flow – Data in Motion



This is what you get. Starting on the left, the machine adapters include OPC-UA, Modbus, MQTT adapter. You can also create your own adapter if these don't service your need.

Sensor data comes into the adapter and is pushed to the Hoover Spillway through the machine gateway.

The spillway is the traffic cop that is configured to send data to a specific River. If you need to run analytics on your data, you can write your own "Edge Processor" module that plugs into the spillway. It's an optional component but will act on your data before it goes to any of the rivers.

Currently, we support 3 types of Rivers: HTTP, WebSocket, and MQTT. The data going through all 3 is batched as a JSON objects going over a secure connection.

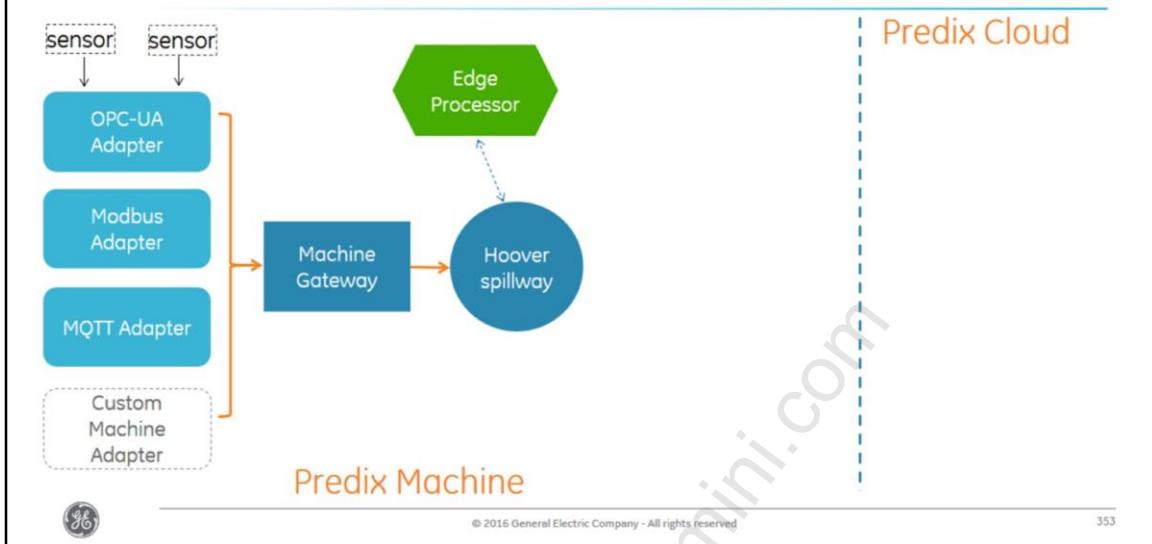
The HTTP river sends the data through HTTPS to an HTTP endpoint in the cloud. The data can be stored in any data management service that makes sense for your use case. The nature of HTTP is to open and close the connection with every request so you may see a measured interval between batches.

The WebSocket river sends data over a websocket. Because websockets stay open until you close them, the pipeline for data going to the cloud is much larger than HTTP. The built-in WebSocket River is designed to connect directly to your timeseries instance in the cloud, therefore you must use the Time Series service to store your data. If you need to use a different data store, you can create your own WebSocket River and configure the spillway to point to it.

If you chose to use the MQTT River, your data will be sent to an MQTT Broker configured to accept the incoming connection request. The Broker can be sitting anywhere, including another installation of Predix Machine. This is how you would daisy-chain your instances of machine together on a plant floor.



Machine Data Flow – Data in Motion



This is what you get. Starting on the left, the machine adapters include OPC-UA, Modbus, MQTT adapter. You can also create your own adapter if these don't service your need.

Sensor data comes into the adapter and is pushed to the Hoover Spillway through the machine gateway.

The spillway is the traffic cop that is configured to send data to a specific River. If you need to run analytics on your data, you can write your own "Edge Processor" module that plugs into the spillway. It's an optional component but will act on your data before it goes to any of the rivers.

Currently, we support 3 types of Rivers: HTTP, WebSocket, and MQTT. The data going through all 3 is batched as a JSON objects going over a secure connection.

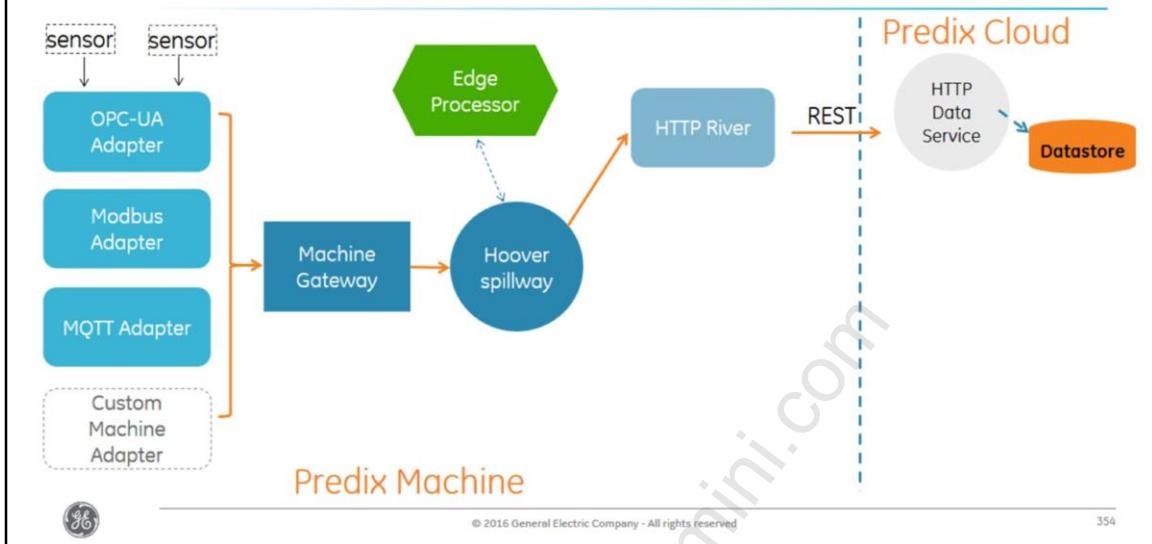
The HTTP river sends the data through HTTPS to an HTTP endpoint in the cloud. The data can be stored in any data management service that makes sense for your use case. The nature of HTTP is to open and close the connection with every request so you may see a measured interval between batches.

The WebSocket river sends data over a websocket. Because websockets stay open until you close them, the pipeline for data going to the cloud is much larger than HTTP. The built-in WebSocket River is designed to connect directly to your timeseries instance in the cloud, therefore you must use the Time Series service to store your data. If you need to use a different data store, you can create your own WebSocket River and configure the spillway to point to it.

If you chose to use the MQTT River, your data will be sent to an MQTT Broker configured to accept the incoming connection request. The Broker can be sitting anywhere, including another installation of Predix Machine. This is how you would daisy-chain your instances of machine together on a plant floor.



Machine Data Flow – Data in Motion



This is what you get. Starting on the left, the machine adapters include OPC-UA, Modbus, MQTT adapter. You can also create your own adapter if these don't service your need.

Sensor data comes into the adapter and is pushed to the Hoover Spillway through the machine gateway.

The spillway is the traffic cop that is configured to send data to a specific River. If you need to run analytics on your data, you can write your own "Edge Processor" module that plugs into the spillway. It's an optional component but will act on your data before it goes to any of the rivers.

Currently, we support 3 types of Rivers: HTTP, WebSocket, and MQTT. The data going through all 3 is batched as a JSON objects going over a secure connection.

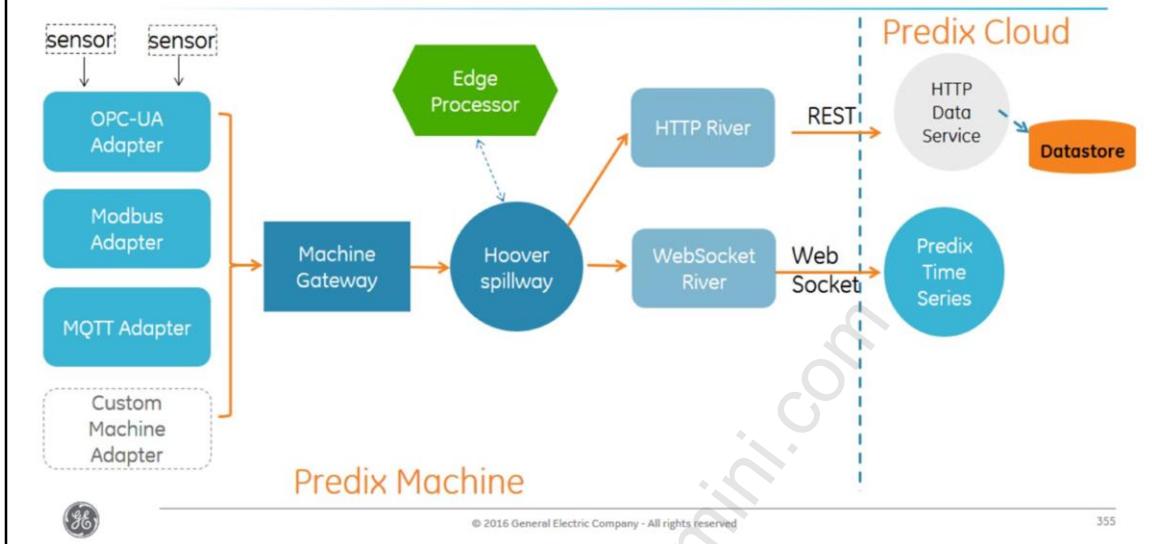
The HTTP river sends the data through HTTPS to an HTTP endpoint in the cloud. The data can be stored in any data management service that makes sense for your use case. The nature of HTTP is to open and close the connection with every request so you may see a measured interval between batches.

The WebSocket river sends data over a websocket. Because websockets stay open until you close them, the pipeline for data going to the cloud is much larger than HTTP. The built-in WebSocket River is designed to connect directly to your timeseries instance in the cloud, therefore you must use the Time Series service to store your data. If you need to use a different data store, you can create your own WebSocket River and configure the spillway to point to it.

If you chose to use the MQTT River, your data will be sent to an MQTT Broker configured to accept the incoming connection request. The Broker can be sitting anywhere, including another installation of Predix Machine. This is how you would daisy-chain your instances of machine together on a plant floor.



Machine Data Flow – Data in Motion



This is what you get. Starting on the left, the machine adapters include OPC-UA, Modbus, MQTT adapter. You can also create your own adapter if these don't service your need.

Sensor data comes into the adapter and is pushed to the Hover Spillway through the machine gateway.

The spillway is the traffic cop that is configured to send data to a specific River. If you need to run analytics on your data, you can write your own "Edge Processor" module that plugs into the spillway. It's an optional component but will act on your data before it goes to any of the rivers.

Currently, we support 3 types of Rivers: HTTP, WebSocket, and MQTT. The data going through all 3 is batched as a JSON objects going over a secure connection.

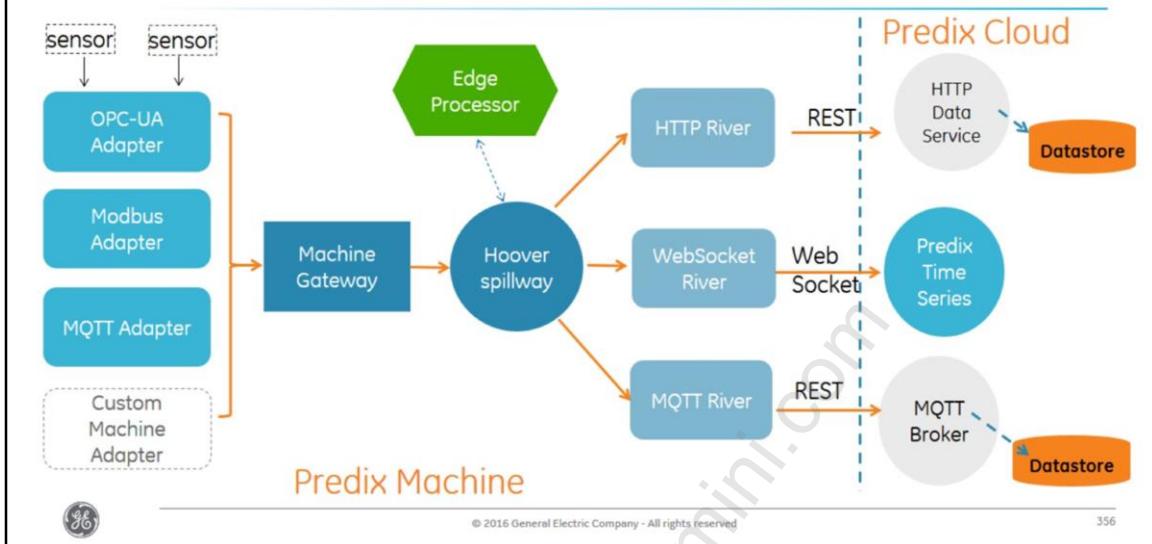
The HTTP river sends the data through HTTPS to an HTTP endpoint in the cloud. The data can be stored in any data management service that makes sense for your use case. The nature of HTTP is to open and close the connection with every request so you may see a measured interval between batches.

The WebSocket river sends data over a websocket. Because websockets stay open until you close them, the pipeline for data going to the cloud is much larger than HTTP. The built-in WebSocket River is designed to connect directly to your timeseries instance in the cloud, therefore you must use the Time Series service to store your data. If you need to use a different data store, you can create your own WebSocket River and configure the spillway to point to it.

If you chose to use the MQTT River, your data will be sent to an MQTT Broker configured to accept the incoming connection request. The Broker can be sitting anywhere, including another installation of Predix Machine. This is how you would daisy-chain your instances of machine together on a plant floor.



Machine Data Flow – Data in Motion



This is what you get. Starting on the left, the machine adapters include OPC-UA, Modbus, MQTT adapter. You can also create your own adapter if these don't service your need.

Sensor data comes into the adapter and is pushed to the Hoover Spillway through the machine gateway.

The spillway is the traffic cop that is configured to send data to a specific River. If you need to run analytics on your data, you can write your own "Edge Processor" module that plugs into the spillway. It's an optional component but will act on your data before it goes to any of the rivers.

Currently, we support 3 types of Rivers: HTTP, WebSocket, and MQTT. The data going through all 3 is batched as a JSON objects going over a secure connection.

The HTTP river sends the data through HTTPS to an HTTP endpoint in the cloud. The data can be stored in any data management service that makes sense for your use case. The nature of HTTP is to open and close the connection with every request so you may see a measured interval between batches.

The WebSocket river sends data over a websocket. Because websockets stay open until you close them, the pipeline for data going to the cloud is much larger than HTTP. The built-in WebSocket River is designed to connect directly to your timeseries instance in the cloud, therefore you must use the Time Series service to store your data. If you need to use a different data store, you can create your own WebSocket River and configure the spillway to point to it.

If you chose to use the MQTT River, your data will be sent to an MQTT Broker configured to accept the incoming connection request. The Broker can be sitting anywhere, including another installation of Predix Machine. This is how you would daisy-chain your instances of machine together on a plant floor.



Predix Machine



Lab 8: Connecting Machines to Predix Cloud

Part II: Moving Data to the Cloud Using HTTP River

- Exercise 1: Start Modbus PLC Simulator



© 2016 General Electric Company - All rights reserved

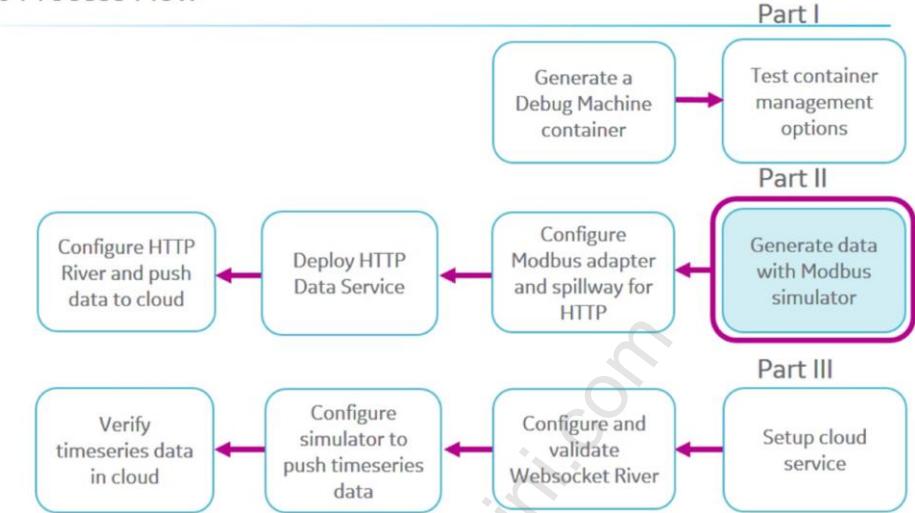
357

This lab will provide some practical experience configuring machine. The first thing you need is data coming in. We use a data simulator, called ModbusPal, that lets you configure assets and sensors to push data over a specific channel.

Exercise 1: In this exercise, you configure a standard Machine container to collect data from Modbus adapter.



Machine Lab Process Flow



© 2016 General Electric Company - All rights reserved

358



Predix Machine

Data Modeling



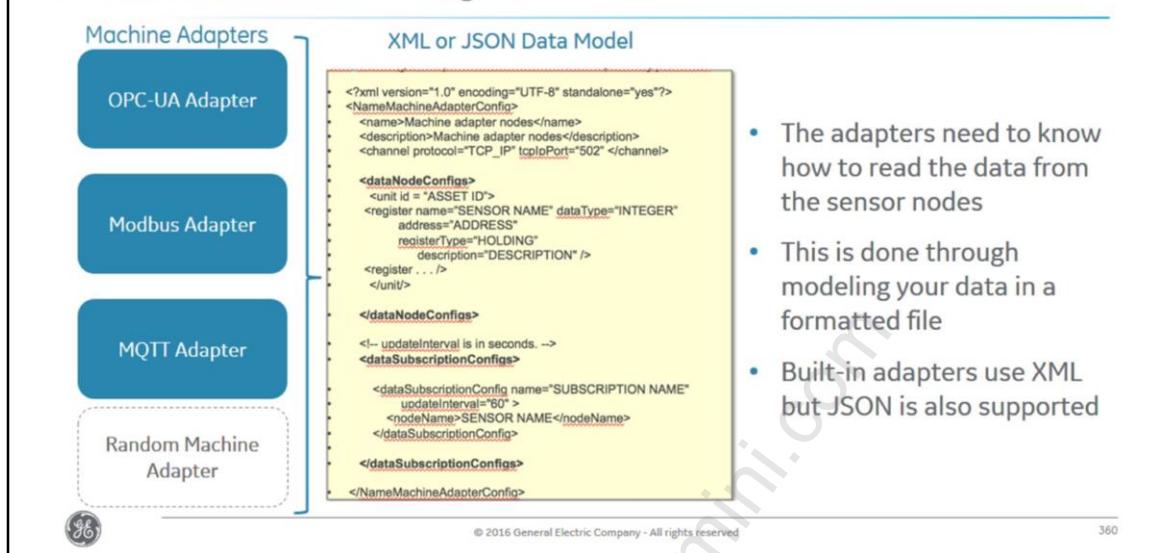
© 2016 General Electric Company - All rights reserved.

359

Now that you have your simulator running, you're pushing data through a dedicated connection but you need to be able to read that data. In order to read that data you need to have a data model in place describing the layout of the data coming in.



Data in Motion: Data Modeling



The component in machine that reads the data coming from sensors is your adapter. The adapter needs a data model to know how to set up and read that data. The data model is in a separate file formatted as either XML or JSON.

Sensor(node) data coming in has very few attributes – usually the sensor name, a timestamp, and the measurement. Here, we're showing what a typical timeseries data object looks like.

A time series uses tags, which are often used to represent sensors (for example, a temperature sensor). A tag consists of one Tag Name (sensor), a Timestamp (time), a Measurement (value). The time series tag can also include quality and one or more attributes (key/value pairs).

You need to configure your adapter to receive the data, by defining a data model to format the data into a xml object.



Data Modeling: Node Configuration

Model to include definition of all sensors for the adapter

Properties:

- Channel | Protocol
- Sensors
 - Name | Data Type | Port (Address) | Description

/com.ge.dspmicro.machineadapter.type.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MachineDataAdapter>
<name>Machine adapter nodes</name>
<description>Machine adapter nodes</description>
<channel protocol="TCP_IP" tcplpAddress="127.0.0.1" tcplpPort="502" >
```

<dataNodeConfigs>

```
<channel protocol="TCP_IP" tcplpAddress="127.0.0.1" tcplpPort="502">
  <unit id="1">
    <register name="Node-1-1" dataType="INTEGER"
      address="10" registerType="HOLDING"
      description="temperature"/>
    <register name="Node-1-2" dataType="DECIMAL"
      address="11" registerType="HOLDING"
      description="pressure"/>
  </unit>
  <unit id="2">
    <register name="Node-2-1" dataType="INTEGER"
      address="20" registerType="HOLDING"
      description="temperature"/>
    <register name="Node-2-2" dataType="INTEGER"
      address="21" registerType="HOLDING"
      description="pressure"/>
  </unit>
</channel>
</dataNodeConfigs>
```

<dataSubscriptionConfigs>

</dataSubscriptionConfig>

</MachineDataAdapterConfig>

© 2016 General Electric Company - All rights reserved

361

If we look at the XML file for the Modbus adapter, you'll see it has tags defined for the channel it's going to read the data over, as well as the format of that data. There is a tag called `<dataNodeConfig>` where you list every sensor this adapter will read from, along with the properties associated with the data. For instance the sensor name, the data type, a description of that data, and the port number (or address) the sensors is sending data through.

In this example we have 2 assets – unit 1 and unit 2 – that are configured for 2 sensors each – Node1-1, Node 1-2, and Node-2-1, Node-2-2.

So that's the first part of the data model.

The XML file describes the adapter connection nodes as well as subscriptions, or interval to collect the data.

```
<channel=TCP/IP or SERIAL>
<channel protocol="TCP_IP" tcplpAddress="127.0.0.1" tcplpPort="502">
<dataNodeConfigs>
<dataSubscriptionConfigs>
com.ge.dspmicro.device.api.xmlmonitor
Interface IXmlListener
```



Data Modeling: Subscription Configuration

Subscription defines the order to read data from sensors

Properties:

- Subscription Name
- Update (read) interval
- Sensors (node) names. Must match name defined in dataNodeConfig section

/com.ge.dspmicro.machineadapter.type.xml



© 2016 General Electric Company - All rights reserved

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MachineDataAdapter>
  <name>Machine adapter nodes</name>
  <description>Machine adapter nodes</description>
  <channel protocol="TCP_IP" tcplpPort="502" />
  <dataNodeConfigs>
  </dataNodeConfigs>
  <dataSubscriptionConfigs>
    <dataSubscriptionConfig name="Temperature_Subscription"
      updateInterval="800" startPointUnit="MINUTES"
      startPointOffset="10">
      <nodeName>Node-2-1</nodeName>
      <nodeName>Node-1-1</nodeName>
    </dataSubscriptionConfig>
    <dataSubscriptionConfig name="Pressure_Subscription"
      updateInterval="3600" startPointUnit="HOURS"
      startPointOffset="600">
      <nodeName>Node-3-2</nodeName>
      <nodeName>Node-4-2</nodeName>
    </dataSubscriptionConfig>
  </dataSubscriptionConfigs>
</MachineDataAdapterConfig>
```

362

The second part of the data model has to do with grouping similar sensors together and reading them by group. You are creating different “subscriptions” and listing the appropriate sensors in each subscription. You create the name of the subscription, and then list the appropriate sensors under that subscription. The data is read in the order listed under the subscription and then sent through as one batch.

In this example we define a subscription called **Temperature_Subscription** that will read from Node-2-1 and Node-1-1 which we presume are temperature sensors. The second subscription is called **Pressure_Subscription** and it reads data from Node-3-2 and Node-4-2 which we presume are sensors that measure pressure.

The XML file describes the adapter connection nodes as well as subscriptions, or interval to collect the data.

```
<channel>TCP/IP or SERIAL</channel>
<channel protocol="TCP_IP" tcplpAddress="127.0.0.1" tcplpPort="502">
<dataNodeConfigs>
<dataSubscriptionConfigs>
  com.ge.dspmicro.device.api.xmlmonitor
  Interface IXmlListener
```



Getting the Data: Sequential Configuration



© 2016 General Electric Company - All rights reserved.

363

So why do you define the sensors in both places? What's the difference between the sequential list of sensors and the ones listed in the subscription section?

First of all Subscriptions are optional. You don't need to define them. Your adapter can read the data based on just the list of nodes in the data node configuration list. Without the subscription it will read the data from sensors in the order they are listed in one batch. If you have 2000 sensors you will get 2000 transactions in one batch.

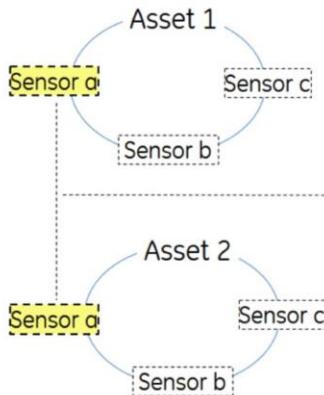
There are 2 ways of getting the data in – Subscription-based OR Site-wide data read.

Let's start with Site-wide

Suppose, you have an asset with sensors generating data. To get that data, implement IMachineGateway and get a handle to your adapter (getMachineAdapters()). Then you get the nodes; start reading the different data points and sending it to store in the cloud. You may need to group "similar" data together to process or store it together in the cloud. That's where subscription-based data comes in.



Getting the Data: Subscription Configuration



`etc/com.ge.dspmicro.machineadapter.type.xml`

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NameMachineAdapterConfig>
<name>Machine adapter nodes</name>
<description>Machine adapter nodes</description>
<channel protocol="TCP_IP" tcpipPort="502" />
<dataNodeConfigs>
<node> name = asset1-sensor a </node>
<node> name = asset1-sensor b </node>
<node> name = asset1-sensor c </node>
<node> name = asset2-sensor a </node>
<node> name = asset2-sensor b </node>
<node> name = asset2-sensor c </node>
</dataNodeConfigs>
<dataSubscriptionConfigs>
<dataSubscriptionConfig name="TEMP" updateInterval="60" >
<nodeName> asset1-sensor a </nodeName>
<nodeName> asset2-sensor a </nodeName>
</dataSubscriptionConfig>
<dataSubscriptionConfig name="PRESSURE" updateInterval="60" >
<nodeName> asset1-sensor b </nodeName>
<nodeName> asset2-sensor b </nodeName>
</dataSubscriptionConfig>
</dataSubscriptionConfigs>
</NameMachineAdapterConfig>
```



© 2016 General Electric Company - All rights reserved

364

When you use a subscription, you are organizing how you are reading the data, and the interval you read that data in. You can group the data from similar sensors under one subscription and send that data through as a single batch. If data coming from those sensors needs to be run through analytics, the data is batched in a way to make analytics more efficient. The analytic doesn't have to sift through 2000 transactions to look for temperature data for instance.

When creating subscriptions, think about what you want to do with that data before it's pushed to the cloud and create your subscriptions around that.

In subscription-based data collection, you are able to group the data from similar sensors across multiple assets.

The grouping is done in the form of a dataSubscription, which is defined along with your data model. In the XML file for the adapter, you create a `<dataSubscriptionConfig>` section and define your "groupings".

Under each grouping, list the sensors (or nodes) you want to collect data from.

You then subscribe to each of these groupings and read data from those sensors.

In the above sample config file, you can see that we are grouping by Temp and Pressure.



Configuring Adapter Properties

```
MachineAdapter.cfg
# Human Readable Adapter Name and Description
com.ge.dspmicro.sample.machineadapter.Name=MachineA
dapter
com.ge.dspmicro.sample.machineadapter.Description=Ada
pter Description

# Polling Interval in Milliseconds [optional]
com.ge.dspmicro.sample.machineadapter.UpdateInterval=
1000
# Number of Nodes [optional]
com.ge.dspmicro.sample.machineadapter.NumberOfNode
s=3
```



© 2016 General Electric Company - All rights reserved.

365

The last piece to configuring your adapter is setting up the runtime properties. Every component, or bundle, in Predix machine has a configuration file that defines the runtime properties. The properties for the adapter include the name and description of the adapter. You can also configure the interval that the adapter reads data from all sensors here, as well as the number of sensors (or nodes) that this adapter will read from.

A better way of configuring these is in the XML data model, and then pointing to the XML file as a runtime property.

There are several ways to configure the properties of the adapter and sensors.
When working with few sensors, you can just add them to your code.
Remember that XML file we used earlier?

Configuring the sensor properties (line update interval and # of nodes) works, but it is pretty basic information and works for small configurations.

However, in a real-world scenario, a better approach is to create a data model in a XML file and configure sensor properties you need. A data model defines what the incoming data looks like. (attributes like adapter name, location of the configuration file for your nodes update interval, and overall number of nodes this adapter is reading data from)

The Modbus adapter in the Predix SDK uses a config file that is set up to work with the machine adapter samples provided.



Configuring Adapter Properties

MachineAdapter.cfg

```
# Human Readable Adapter Name and Description  
com.ge.dspmicro.sample.machineadapter.Name=MachineAdapter  
  
com.ge.dspmicro.sample.machineadapter.Description=Adapter Description  
  
# Configuration File for Data Model [optional]  
com.ge.dspmicro.machineadapter.configFile=etc/com.  
ge.dspmicro.machinedataadapter-0.xml
```

Modbus-adapter.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<NameMachineAdapterConfig>  
<name>Machine adapter nodes</name>  
<description>Machine adapter nodes</description>  
<channel protocol="TCP_IP" tcpIpPort="502" />  
  
<dataNodeConfigs>  
<unit id = "ASSET ID">  
<register name="SENSOR NAME" dataType="INTEGER"  
address="ADDRESS"  
registerType="HOLDING"  
description="DESCRIPTION" />  
<register ... />  
</unit/>  
  
</dataNodeConfigs>  
<!-- updateInterval is in seconds. -->  
<dataSubscriptionConfigs>  
  
<dataSubscriptionConfig name="SUBSCRIPTION NAME"  
updateInterval="60" >  
<nodeName>SENSOR NAME</nodeName>  
</dataSubscriptionConfig>  
</dataSubscriptionConfigs>  
  
</NameMachineAdapterConfig>
```



© 2016 General Electric Company - All rights reserved

366

The last piece to configuring your adapter is setting up the runtime properties. Every component, or bundle, in Predix machine has a configuration file that defines the runtime properties. The properties for the adapter include the name and description of the adapter. You can also configure the interval that the adapter reads data from all sensors here, as well as the number of sensors (or nodes) that this adapter will read from.

A better way of configuring these is in the XML data model, and then pointing to the XML file as a runtime property.

There are several ways to configure the properties of the adapter and sensors.
When working with few sensors, you can just add them to your code.
Remember that XML file we used earlier?

Configuring the sensor properties (line update interval and # of nodes) works, but it is pretty basic information and works for small configurations.

However, in a real-world scenario, a better approach is to create a data model in a XML file and configure sensor properties you need. A data model defines what the incoming data looks like. (attributes like adapter name, location of the configuration file for your nodes update interval, and overall number of nodes this adapter is reading data from)

The Modbus adapter in the Predix SDK uses a config file that is set up to work with the machine adapter samples provided.



Predix Machine



Lab 8: Connecting Machines to Predix Cloud

Part II: Moving Data to the Cloud Using HTTP River

- [Exercise 2: Configure Modbus Adaptor in Machine Services](#)
- [Exercise 3: Configure a Spillway for HTTP](#)
- [Exercise 4: Import HTTP Data Service Sample from Predix SDK](#)
- [Exercise 5: Configure HTTP River](#)



© 2016 General Electric Company - All rights reserved.

367

This lab will provide some practical experience configuring machine. The first thing you need is data coming in. We use a data simulator, called ModbusPal, that lets you configure assets and sensors to push data over a specific channel.

Exercise 2: Configure Modbus adaptor in machine services

Predix Machine supports Modbus through the Modbus adapter bundle. This bundle must be configured to work with a given Modbus device.

Exercise 3: Configure a spillway for HTTP

The Data River service allows data to be passed through spillways for processing. In this example, we will set up a simple subscription-based spillway to listen to the Modbus adapter.

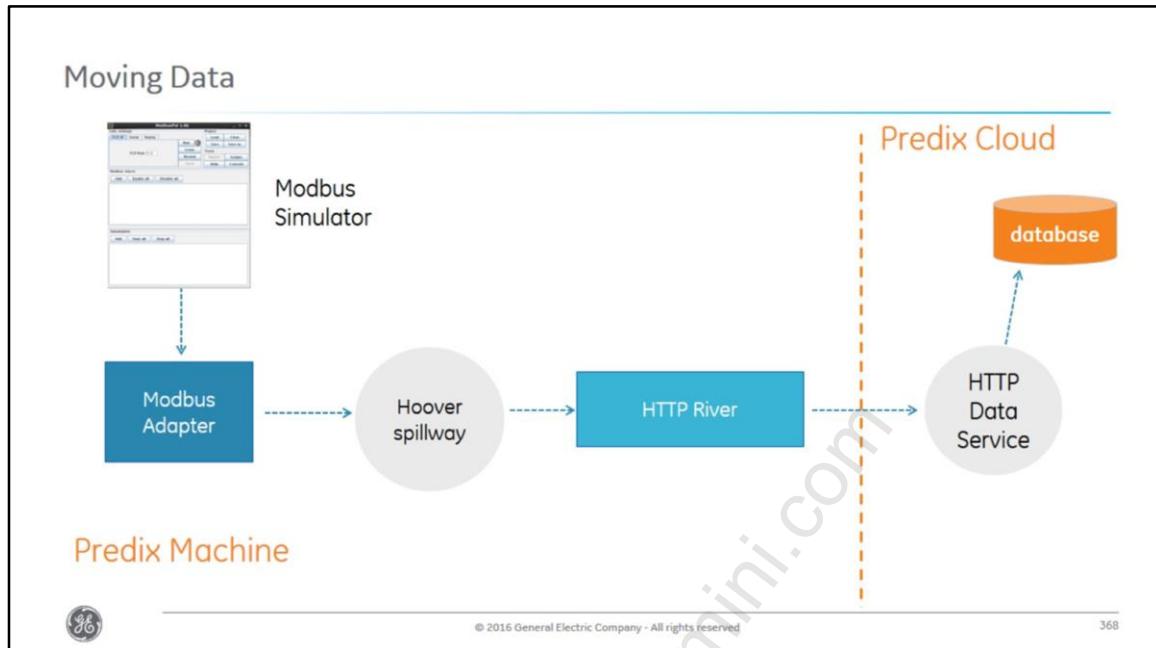
Exercise 4: Import HTTP data service sample from Predix SDK

In this exercise you import and build the HTTP Data Service sample that runs in the cloud. This service is the endpoint for the HTTP River sending data from Predix Machine.

Exercise 5: Configure HTTP river

Configure the HTTP Machine Services and start the flow of data.





In the next lab you will use the Modbus simulator that you started. It's sending data to the Modbus Adapter but the adapter doesn't know how to read that data in without a data model.

You'll need to configure the data model (or XML file) for the Modbus adapter.

The spillway is by default configured to push data to the HTTP River but the HTTP River needs to know where to push the data to.

You'll pull a sample app from the machine SDK and push it to the cloud. This app is already set up to write to a postgres instance so your data will go into the postgres service you set up earlier.

Start the Modbus simulator to generate simulated sensor data.

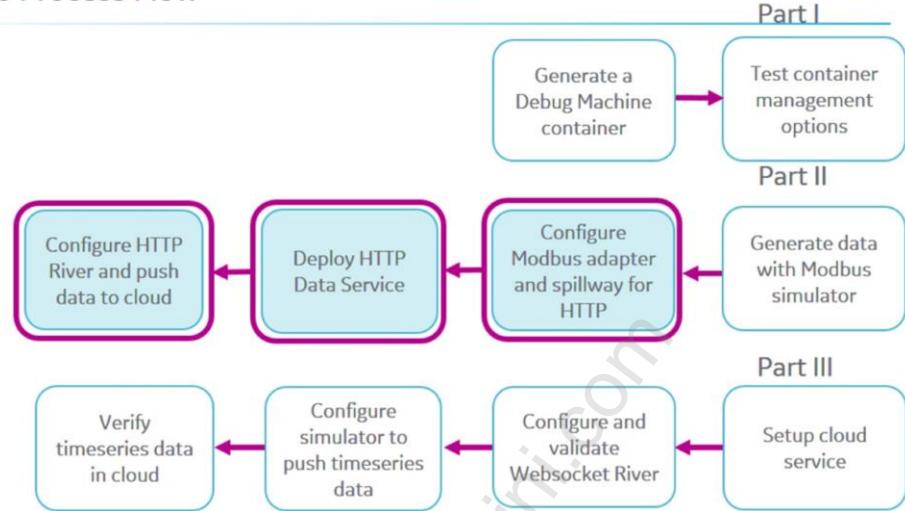
Configure your machine container to collect data from the Modbus adapter.

Configure a spillway to process the data and forward it to the HTTP river.

Configure the HTTP river to start the flow of data to data store(cloud).



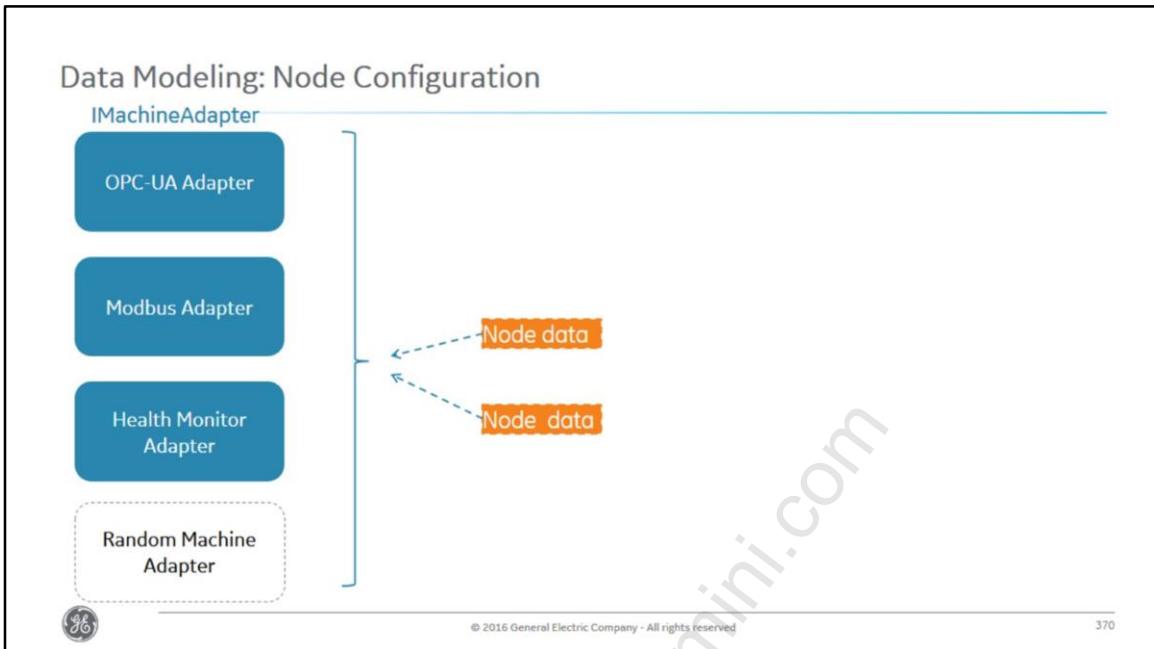
Machine Lab Process Flow



© 2016 General Electric Company - All rights reserved

369





If we look at the XML file for the Modbus adapter, you'll see it has tags defined for the channel it's going to read the data over, as well as the format of that data. There is a tag called <dataNodeConfig> where you list every sensor this adapter will read from, along with the properties associated with the data. For instance the sensor name, the data type, a description of that data, and the port number (or address) the sensors is sending data through.

In this example we have 2 assets – unit 1 and unit 2 – that are configured for 2 sensors each – Node1-1, Node 1-2, and Node-2-1, Node-2-2.

So that's the first part of the data model.

The XML file describes the adapter connection nodes as well as subscriptions, or interval to collect the data.

```

<channel>TCP/IP or SERIAL</channel>
<channel protocol="TCP_IP" tcplpAddress="127.0.0.1" tcplpPort="502">
<dataNodeConfigs>
<dataSubscriptionConfigs>
com.ge.dsmpmicro.device.api.xmlmonitor
Interface IXmlListener
    
```



Data Modeling: Node Configuration

IMachineAdapter

- OPC-UA Adapter
- Modbus Adapter
- Health Monitor Adapter
- Random Machine Adapter

Node data

Node data

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NameMachineAdapterConfig>
<name>Machine adapter nodes</name>
<description>Machine adapter nodes</description>
<channel protocol="TCP_IP" tcplpAddress="127.0.0.1" tcplpPort="502"></channel>

<dataNodeConfigs>
<unit id = "ASSET ID">
<register name="SENSOR NAME" dataType="INTEGER">
<address>ADDRESS</address>
<registerType>HOLDING</registerType>
<description>DESCRIPTION</description>
<register ... />
</register/>
</unit/>
</dataNodeConfigs>
<!-- updateInterval is in seconds. -->
<dataSubscriptionConfigs>
<dataSubscriptionConfig name="SUBSCRIPTION NAME" updateinterval="60">
<nodeName>SENSOR NAME</nodeName>
</dataSubscriptionConfig>
</dataSubscriptionConfigs>
</NameMachineAdapterConfig>
```

© 2016 General Electric Company - All rights reserved.

371

If we look at the XML file for the Modbus adapter, you'll see it has tags defined for the channel it's going to read the data over, as well as the format of that data. There is a tag called <dataNodeConfig> where you list every sensor this adapter will read from, along with the properties associated with the data. For instance the sensor name, the data type, a description of that data, and the port number (or address) the sensors is sending data through.

In this example we have 2 assets – unit 1 and unit 2 – that are configured for 2 sensors each – Node1-1, Node 1-2, and Node-2-1, Node-2-2.

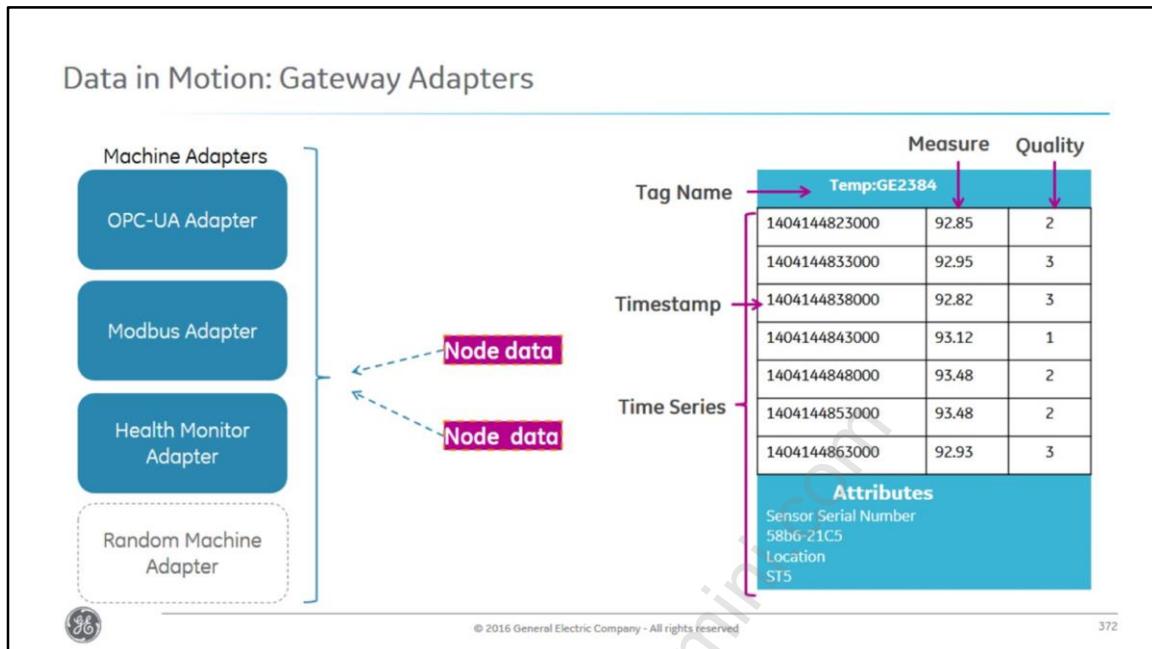
So that's the first part of the data model.

The XML file describes the adapter connection nodes as well as subscriptions, or interval to collect the data.

```
<channel>TCP/IP or SERIAL</channel>
<channel protocol="TCP_IP" tcplpAddress="127.0.0.1" tcplpPort="502">
<dataNodeConfigs>
<dataSubscriptionConfigs>
com.ge.dspmicro.device.api.xmlmonitor
Interface IXmlListener
```



Data in Motion: Gateway Adapters



We've seen that every adapter requires a data model in an XML or JSON file. Once the data is read in by the adapter it is formatted into an object and sent through. Aside from the data value itself, properties included in that object include the timestamp, the quality of service (QoS) value, and the sensor (or node) name, all of which are required by timeseries.

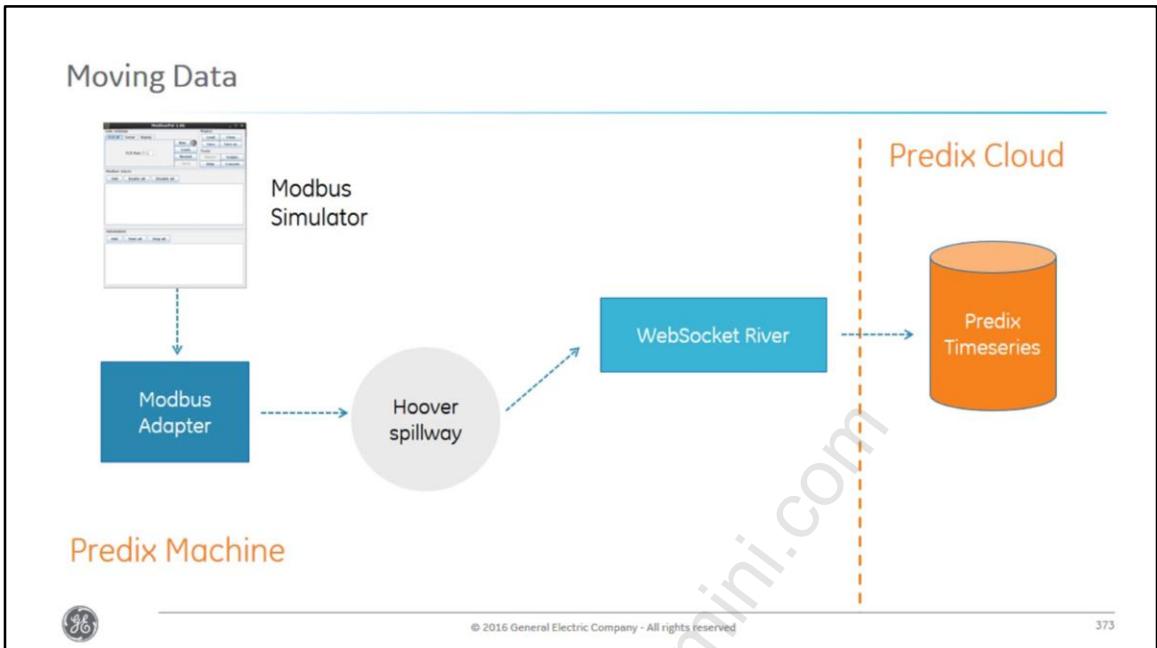
If you look at machine data, every datapoint is associated with a TAG name. With machine, the TAG Name comes from the data model. The TAG Name is the NODE NAME in the XML data model file. The adapter provides the timestamp as well as the QoS (quality of service) value for every data point. The QoS always defaults to "GOOD". If you need to validate that the data is actually GOOD, use an edge analytic and update the QoS value there before sending it to timeseries.

Sensor(node) data coming in has very few attributes – usually the sensor name, a timestamp, and the measurement. Here, we're showing what a typical timeseries data object looks like.

A time series uses tags, which are often used to represent sensors (for example, a temperature sensor). A tag consists of one Tag Name (sensor), a Timestamp (time), a Measurement (value). The time series tag can also include quality and one or more attributes (key/value pairs).

You need to configure your adapter to receive the data, by defining a data model to format the data into a xml object.





- Reconfigure your Predix machine to push data to Websocket river
- Validate the Websocket river
- Configure the Modbus simulator to generate Timeseries data
- Push the data to cloud
- Validate the data.



Predix Machine

</Lab>

Lab 8: Connecting Machines to Predix Cloud

Part III: Connect to the Cloud with WebSocket River

- Exercise 1: Cloud Service Setup
- Exercise 2: Validate Websocket River
- Exercise 3: Validating Data in Time Series



© 2016 General Electric Company - All rights reserved.

374

Now that you've configured machine to use the HTTP River and push data to a postgres instance in the cloud let's configure machine to push data to a timeseries instance. In order to do this you must use the WebSocket River.

Exercise 1: Cloud service setup

In this exercise you reconfigure Predix machine to push data over WebSocket River.

Exercise 2: Validate Websocket river

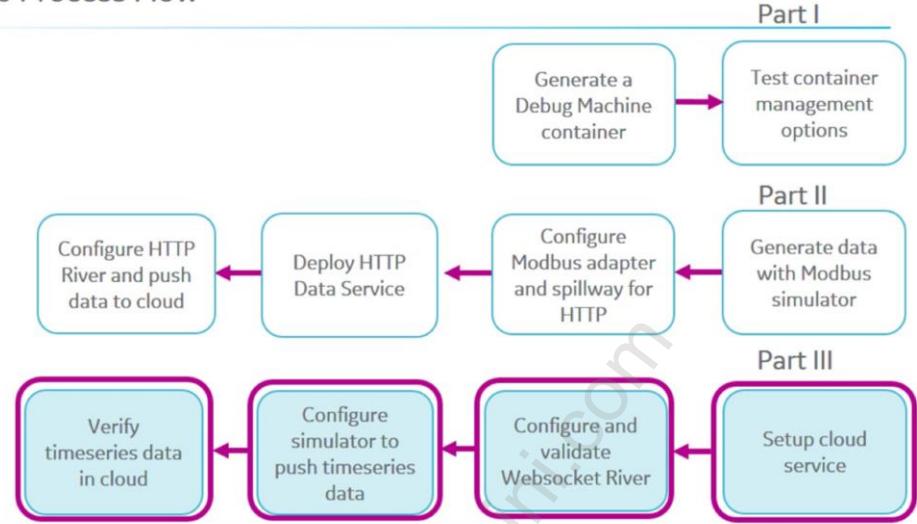
In this exercise, you verify that the WebSocket River can connect to Predix Cloud.

Exercise 3: Validating data in Time Series

In this exercise you will query the data coming into timeseries through websocket river.



Machine Lab Process Flow



© 2016 General Electric Company - All rights reserved.

375



Predix Machine

Edge Manager



© 2016 General Electric Company - All rights reserved

376



Edge Manager: Administrator Tool

Audience is administrator (not developer)

- 1 of 3 views identified in configuration

Used for software provisioning only

- Not monitoring sensor data or thresholds to determine faulty unit

Register or enroll physical hardware running machine

- Based on MAC address of device running machine

S/W updates are scheduled for a specific time



© 2016 General Electric Company - All rights reserved.

377

Audience is administrator (not developer):

1 of 3 views identified in configuration. Views of data are different depending on who you are (Admin, Technician, Operator)

Used for software provisioning only:

Not monitoring sensor data or thresholds to determine faulty unit.

Still need separate UI running analytics on data for that.

- Register or enroll physical hardware running machine (not sensor).
- Based on MAC address of device running machine
- Site-wide (or tenant-based) UAA instance that each gateway is enrolled with
- S/W updates are scheduled for a specific time and updates are pulled over by machine.



Module Summary: Predix Machine

- Predix Machine is a software solution that makes industrial assets “Internet- Ready”
- Predix machine is hardware agnostic
- Predix Machine enables asset data transfer to Predix cloud for Processing and storage via HTTP / WebSocket rivers.
- Sensors need a data model to know how to set up data (XML or JSON format)
- Predix Machine applications may cache data locally when connections are down



© 2016 General Electric Company - All rights reserved

378



Course Feedback

Student Survey Link: www.MetricsThatMatter.com/FLaneus05

- Click on the dropdown arrow
- Find your name, date and instructor's name
- Start completing the feedback form

Note: Your class will not be visible until the last day on instruction.



© 2016 General Electric Company - All rights reserved

379





Thank You

