Get Started for Predix Developers

Contents

Predix Overview

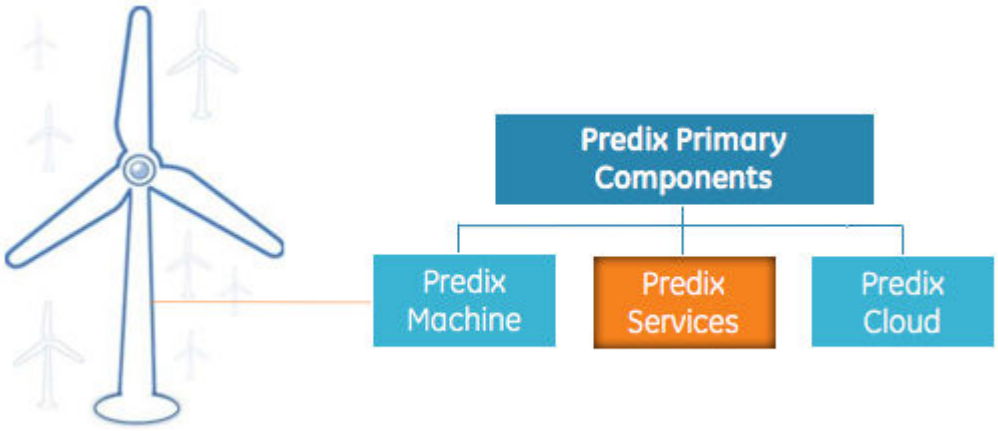
What is Predix Platform?

The Predix platform is a cloud-based Platform-as-a-Service (PaaS) for the Industrial Internet.

As it connects machines, data, people, and other assets, the Predix platform uses leading technologies for distributed computing, big-data analytics, asset data management, and machine-to-machine communication. The platform provides a wide range of industrial microservices that enable businesses to increase productivity, and it provides the following benefits:

- Enables the rapid development of industrial applications.
- Minimizes developer involvement in managing scale and hardware.
- Promotes quick response to customer requirements.
- Provides customers a single point of control for the assets they own.
- Serves as a foundation for an ecosystem of companies and developers that support the Industrial Internet.

To understand the Predix platform, let's start with an industrial asset, such as a wind turbine, and see how it is connected to the primary Predix components: Predix Machine, Predix cloud, and Predix services.



Predix Machine

Predix Machine is the software layer responsible for collecting data from industrial assets and pushing it to the Predix cloud, as well as running local applications, like edge analytics. Predix Machine is installed on gateways, industrial controllers, and sensors.

Predix Services

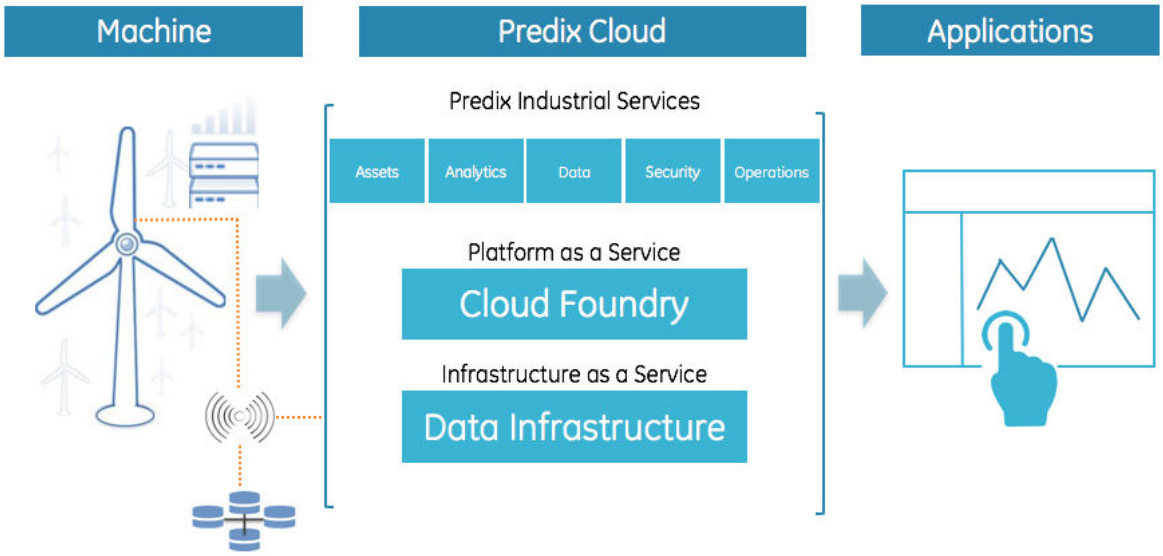
Predix provides industrial services that developers can use to build, test, and run industrial internet applications. It also provides a microservices marketplace where developers can publish their own services as well as consume services from third parties. The Predix platform enables our customers and partners to optimize their industrial business processes.

Predix Cloud

The Predix cloud is a global, secure cloud infrastructure that is optimized for industrial workloads and meets strict regulatory standards for such industries as healthcare and aviation.

Predix Platform Architecture

One example that illustrates the architecture of the Predix platform is a wind farm application that collects data from turbines and pushes it to the cloud.



The wind turbine and the turbine farm sit on the "edge" of the Predix platform. Using Predix services, you can receive and analyze the data from the wind turbine sensors. You can also monitor and optimize the operation of the turbine to gain maximum value of this asset. This enables you to detect anomalies and help predict outages before they happen, improving wind farm reliability, optimizing just-in-time maintenance, and reducing turbine downtime.

Predix Machine is a hardware and software solution that uses data collected from the sensors and uses edge analytics to monitor the status of industrial assets. If something out of the ordinary is detected, Predix Machine can shut down a turbine before damage occurs. With Predix Machine, data scientists can store and analyze data across the wind farm. They can look for trends over time, identify new patterns, create new edge analytics, and push that information back out to all of the wind turbines. Predix connectivity allows Predix Machine to talk to the cloud even when the internet is down. This offline support capability provides applications with constant data access, even when the network is not available.

Application developers can use the industrial services on the Predix cloud to build, test, and deploy Industrial Internet applications. This custom-built cloud data infrastructure has enhanced security controls and advanced data processing and networking capabilities. From improved analytics, real-time asset optimization, or predictive maintenance, the Predix platform is designed to support the continuous improvement of industrial business processes.

Predix Microservices

The Predix platform employs a Cloud Foundry-based architecture that supports microservice use and delivery. Microservices deliver functionality as a set of very small, granular, independent collaborating services.

Predix platform microservices simplify the operation and management of deployed applications, IT operational complexity, solution integration, and solution management. Once a solution has been deployed, updates are simpler and more efficient, eliminating code recompilation and streamlining operations.

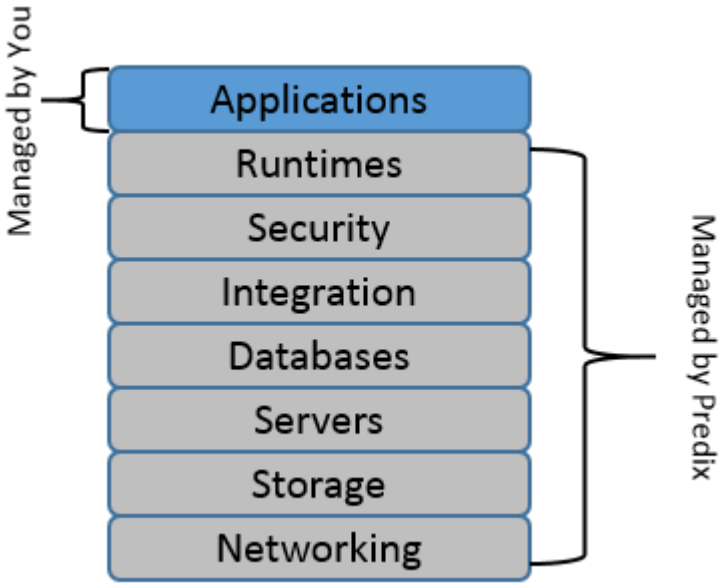
The Predix platform provides a wide range of industrial microservices to create predictivity solutions which enable businesses to increase productivity through asset performance management, operations optimizations, asset modeling, data ingestion, data storage and manipulation, and security for all back-end applications.

Predix and Cloud Foundry

Predix is built on Cloud Foundry, an open source Platform-as-a-Service (PaaS) implementation that helps reduce your development costs.

Cloud Foundry supports the software lifecycle from initial development through each testing stage to deployment. Cloud Foundry's advantage is its strong support of continuous delivery (CD) software strategies. Cloud Foundry provides the following benefits:

- Application lifecycle management
- Centralized management of applications
- Distributed environment
- Easy maintenance



See also Why Does Predix Use Cloud Foundry?

What You Get When You Register for Predix

When you register for a Predix account, a Predix administrator creates a Cloud Foundry user account for you and gives you a Cloud Foundry org and space. An individual account gives you one org and one space within that org. An Enterprise account gives you one org with one or more spaces.

What is a Cloud Foundry Org?

Cloud Foundry describes an org as a development account that an individual or multiple collaborators can own and use. All collaborators access an org with user accounts. Collaborators in an org share a resource quota plan, applications, services availability, and custom domains. An org allows you to create a multi-tenant environment.

All orgs include a quota plan, and all activities within the org are counted in the quota plan for the org. A quota plan tracks all activities within the org as well as data such as memory that is used by applications, the services they use, and disk usage.

An org always includes at least one space, but it can contain multiple spaces. You can define the org any way you want. For example, your org can correspond to a business division.

What is a Cloud Foundry User Account?

Cloud Foundry describes a user account as an account that represents an individual person within the context of a Cloud Foundry installation. A user can have different roles in different spaces within an org, governing the level and type of access they have within that space.

What is a Cloud Foundry Space?

Cloud Foundry describes a space as a shared location for application development, deployment, and maintenance. Each space role applies only to a particular space.

Each application and service is scoped to a space. Spaces are often defined by logical environment, such as development, testing, staging, or production.

What is a Cloud Foundry System Buildpack?

Cloud Foundry includes system buildpacks that provide framework and runtime support for your applications. For a complete list of Cloud Foundry buildpacks, see https://docs.cloudfoundry.org/buildpacks/.

Predix Application Development Best Practices

Follow these guidelines when designing applications to run on Cloud Foundry.

- Follow the twelve-factor app methodology when designing your application. See http://12factor.net/.
- Applications running on Cloud Foundry should not write files to the local file system because local file system storage is short-lived. In addition, instances of the same application do not share a local file system.
- Be aware of port limitations. For example, HTTP requests arrive on ports 80 and 443, and Cloud Foundry requires a channel for TCP/WebSocket traffic (port 4443 is assigned by default).
- When deploying your application to Cloud Foundry, ignore unnecessary files that could slow the deployment.
- Run multiple instances of your application to keep it available while undergoing Cloud Foundry upgrade processes.

See also https://docs.cloudfoundry.org/devguide/deploy-apps/prepare-to-deploy.html.

Additional Information

- Learning Predix Guide
- Predix CLI Guide
- Hello World Guide

Predix Development Environment Setup

Task Roadmap: Predix Development Environment Setup

Register for a Predix account, install the required software, and set up access for your environment.

#	Task	Description
1	Register for a Predix Account	When you register for a Predix account, a Cloud Foundry user account is created and you are given a Cloud Foundry org and space. For more information, see Registering for a Predix Account on page 5.
2	Install the Cloud Foundry Command Line Tool	You can use the Cloud Foundry CLI to deploy and manage applications and services. For more information, see Installing the Cloud Foundry Command Line Tool on page 6.
3	Install the Application Development Tools	Before you can develop applications on the Predix platform, you must install the appropriate software on your local machine. For more information, see Common Cloud Development Tools on page 6.
4	Define Proxy Connections	You may need to configure your proxy settings to access remote resources. For more information, see Defining Proxy Connections to Remote Resources on page 7.
5	Set up Access to Predix GitHub Repository	You can set up access to the PredixDev repository to access the sample Predix applications and sample code. For more information, see Accessing the PredixDev GitHub Repository on page 8.
6	Install and Configure Maven	You can use Maven to work with the Predix sample applications and code libraries to build Java projects. For more information, see Installing the Maven Build Tool on page 9.

Registering for a Predix Account

When you register for a Predix account, a user account is created and you are given an org and space.

About This Task

An individual account gives you one org and one space within that org. An Enterprise account gives you one org with one or more spaces. Individual users can add additional spaces to an org assigned to them. Enterprise users can add additional users and spaces to the org assigned to them using the Predix.io user interface.

Procedure

1. To register as an individual user, go to http://predix.io and click **Sign Up**.

   If you are an enterprise user, contact Predix Customer Support at gedigital@ge.com for help setting up your account.
2. Select your **Country**, choose **Personal** or **Company** account, enter your **Email Address**, and click **Continue**.
3. Enter your information on the Sign-up form and click **Verify Information**.

   A 4-digit pin is delivered to the email address you provided.
4. Enter the 4-digit PIN sent to your Work Phone or Work Email.
5. Check your email for details about registration.

Installing the Cloud Foundry Command Line Tool

Download and install the Cloud Foundry CLI to deploy and manage applications and services.

Procedure

Download the CLI from https://github.com/cloudfoundry/cli#downloads and follow the installation instructions.

For information about Cloud Foundry CLI commands, see https://docs.cloudfoundry.org/cf-cli/.

Common Cloud Development Tools

You can use many common developer tools to create applications on the Predix platform.

Your Predix.io account gives you access to various resources, such as the PredixDev GitHub and Artifactory, which contain sample code. The following table shows the most common tools for Predix developers.

Table 1: Common Tools for Cloud Development

Software	Version	Download
Android Studio	Latest	https://developer.android.com/sdk/index.html
Bower	Latest stable version	https://bower.io/
Eclipse	Latest	https://eclipse.org/users/.
Git	Latest	https://git-scm.com/downloads
Java SE Development Kit (JDK)	8	https://www.oracle.com/downloads/index.html
Maven	Latest	If you are developing with Java, you can use Maven to manage and organize dependencies for your project. https://maven.apache.org/download.cgi

Software	Version	Download
Node.js	Latest	https://nodejs.org/.
XCode	7.3.1	http://adcdownload.apple.com/Developer_Tools/Xcode_7.3.1/Xcode_7.3.1.dmg

Table 2: Common Tools for Windows

Software	Version	Download
Cloud Foundry extensions for Visual Studio	Latest	Enables you to publish applications from Visual Studio directly to a Cloud Foundry deployment. https://visualstudiogallery.msdn.microsoft.com/4cad4d95-099c-449e-9d90-7d4da5c4a0c0.
Cygwin	Latest	Allows porting of many UNIX programs without the need for extensive changes to the source code. This includes configuring and building most of the available GNU or BSD software. They can be used from one of the provided UNIX shells like `bash`, `tcsh` or `zsh`. When you install Cygwin, select the curl libs installation so that you can perform curl commands. https://cygwin.com/install.html.
Visual Studio Community	Latest	https://www.visualstudio.com/.

Defining Proxy Connections to Remote Resources

If traffic between your corporate network and the internet is monitored, access to certain tools (cURL, Maven, Git, Eclipse, and so on) may be blocked by the proxy.

About This Task

You may need to configure your proxy settings to access remote resources. The values for your proxy settings vary depending on your location and network configuration. Contact your IT administrator for the correct proxy settings.

Procedure

- For Windows:
  1. On the **Start** menu, right-click **Computer** > **Properties**.
  2. Click **Advanced system settings**.
  3. Click the **Advanced** tab.

4. Click **Environment Variables**.
5. Click **New** and enter the **Variable Name** and **Variable Value** for each variable.

   Set these values:

Variable Name	Variable Value
HTTP_PROXY	<host>:<port>
HTTPS_PROXY	<host>:<port>
http_proxy	<host>:<port>
https_proxy	<host>:<port>
no_proxy	<domain>

   Note: HTTP_PROXY and HTTPS_PROXY variables are case-insensitive on Windows.

   To set the environment variables for the current session, enter the following from a command prompt:

```
set HTTP_PROXY=http://<host>:<port>
set HTTPS_PROXY=http://<host>:<port>
set no_proxy=<domain>
```

   For example:

```
set HTTP_PROXY=http://proxy.mycompany.com:8080
set HTTPS_PROXY=http://proxy.mycompany.com:8080
set no_proxy=mycompany.com
```

- For Mac OS:
  - Define system proxy connections to remote resources by adding the following entries to the ~/.profile, ~/.bash_profile or ~/.bashrc file.

  For example, MacOS settings should reflect the following changes to the ~/.bash_profile file:

```
export HTTP_PROXY=http://<host>:<port>
export HTTPS_PROXY=http://<host>:<port>
export http_proxy=http://<host>:<port>
export https_proxy=http://<host>:<port>
export no_proxy=<domain>
```

- **Additional Information:**

  Dev Tools: Define network proxy settings

Accessing the PredixDev GitHub Repository

Sample applications and sample code are available in the PredixDev repository.

Procedure

1. Download and install the latest version of the Git client for your OS from https://git-scm.com/downloads.
2. Verify your version of Git:

```
git --version
```

3. Navigate to https://github.com/predixdev to verify your access to the developer repositories.

Installing the Maven Build Tool

You can use Maven to work with the Predix sample applications and code libraries to build Java projects.

Procedure

1. Download and install the latest version of Maven from https://maven.apache.org/download.cgi.
2. Verify your Maven installation. At the command prompt, enter:

```
mvn --version
```

Related tasks

Defining Your Maven Project and Dependencies on page 9

Maven uses the `pom.xml` file to configure information that applies to the project.

Defining Predix Platform Artifactory Access on page 10

If you need Predix platform artifacts for your application development, you can set up your environment to access the Predix platform Artifactory.

Adding the Predix Artifactory Repository to Your Project on page 11

Define the repository in your POM file.

Defining Your Maven Project and Dependencies

Maven uses the `pom.xml` file to configure information that applies to the project.

About This Task

Each Java project that uses Maven places a `pom.xml` file in the root directory of the project. Maven uses the information specified in the `pom.xml` to build the project. The `pom.xml` file contains project information such as name, dependencies, version, and packaging instructions.

Procedure

1. Define your project `groupid`, `artifactid`, and `version`.

   The `groupId` element allows you to group several projects into the same namespace. The `artifactId` element identifies the name of the project, in this case, `predix-microservice-template-cf-jsr`. The `version` element sets a version for the project.

```
<groupId>com.ge.predix.solsvc</groupId>
<artifactId>predix-microservice-cf-jsr</artifactId>
<version>1.1.4</version>
```

2. Define your project dependencies.

   The following `dependency` element defines a dependency to a Spring Framework Starter Web project. This means the Java project gets to call code from that project or it leverages functionality provided by that project.

```
<dependencies>
     <dependency>
         <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-web</artifactId>
        <version>1.2.3.RELEASE</version>
    </dependency>
</dependencies>
```

Defining Predix Platform Artifactory Access

If you need Predix platform artifacts for your application development, you can set up your environment to access the Predix platform Artifactory.

About This Task

An Artifactory is a binary repository manager that provides version control (and more) for your binary artifacts (such as `jar` and `war` files). Maven uses a settings file to define proxies and define the Predix platform Artifactory.

Procedure

1. Update your proxy settings in the `settings.xml` file to access the Artifactory.

   For example, if you are in the GE network, you can update the proxy settings as follows:

   ```
   <proxies>
      <proxy>
         <id>optional</id>
         <active>true</active>
         <protocol>http</protocol>
         <host><GE_proxy_server></host>
         <port>8080</port>
         <nonProxyHosts>*.ge.com</nonProxyHosts>
      </proxy>
   </proxies>
   ```

   Note: This step is required only if you are behind a proxy.
2. Update your Maven settings to use the Predix platform Artifactory:

   ```
   <server>
       <id>artifactory.external</id>
       <username><predix-user-name></username>
       <password><predix-password></password>
   </server>
   ```

   where `<predix-user-name>` and `<predix-password>` are your Predix.io username and password.

   You can encrypt your password instead of specifying in plain text. For more information, see Maven's Password Encryption.

Adding the Predix Artifactory Repository to Your Project

Define the repository in your POM file.

Procedure

Add the repository to your application `pom.xml` file.

```
<repositories>
      <repository>
         <id>artifactory.external</id>
         <name>GE external repository</name>
         <url>https://artifactory.predix.io/artifactory/PREDIX-EXT</
url>
      </repository>
</repositories>
```

Deploying an Application to Cloud Foundry

Creating and Deploying a Simple Web App to Cloud Foundry

To create and deploy a simple web application, you can clone a copy of the Predix Hello World web application and deploy it to Cloud Foundry.

Procedure

1. Use the Cloud Foundry CLI to log into Cloud Foundry.

   ```
   cf login -a <API_Endpoint>
   ```

   Note: If you are a GE employee, you must use the `cf login --sso` command to log into Cloud Foundry. After you enter your SSO, you will receive a one-time passcode URL. Copy this URL and paste it in a browser to retrieve your one-time passcode. Use this code with the `cf` command to complete the CF login process.

   Depending on your Predix.io registration, the value of *<API_Endpoint>* is one of the following:

   - Predix US-West
     https://api.system.aws-usw02-pr.ice.predix.io
   - Predix US-East
     https://api.system.asv-pr.ice.predix.io
   - Predix Europe
     https://api.system.aws-eu-central-1-pr.ice.predix.io

   For example,

   ```
   cf login -a https://api.system.aws-usw02-pr.ice.predix.io
   ```

2. Clone a copy of the Predix Hello World web app to create a Predix Hello World project. At the command prompt, enter:

   ```
   git clone https://github.com/PredixDev/Predix-HelloWorld-WebApp
   ```

   The command produces the following output:

   ```
   Cloning into
       'Predix-HelloWorld-WebApp'...
   remote: Counting objects: 7,
       done.
   remote: Compressing objects:
       100% (5/5), done.
   remote: Total 7 (delta 0),
       reused 7 (delta 0), pack-reused 0
   Unpacking objects: 100% (7/7),
       done.
   Checking connectivity... done.
   ```

3. Edit the `manifest.yml` file in the `Predix-HelloWorld-WebApp` directory to update the name of the web app to `Predix-HelloWorld-WebApp-`*<YourAppName>*. For example:

   ```
   #
   applications:
     - name:
   ```

```
      Predix-HelloWorld-WebApp-<YourAppName>
    buildpack: predix_openresty_buildpack
    #path: dist
    memory: 64M
    stack: cflinuxfs2
```

4. Make sure that you are in the `Predix-HelloWorld-WebApp` directory. From the directory, push your Predix web application to Cloud Foundry. At the command prompt, enter:

```
cf push
```

5. See your web app listed in your Cloud Foundry space:

```
cf apps
```

6. Enter your web app URL in a browser, including your app name, using HTTPS. For example:

```
 https://Predix-HelloWorld-WebApp-<YourAppName>.run.aws-usw02-
pr.ice.predix.io
```

   Your Predix Hello World web app displays in the web browser.

Related concepts

Create Modern Web Applications

Deploying Applications Using a Manifest File

You can use a manifest file to provide information to Cloud Foundry about your application, such as what services to use when you push the application.

About This Task

While using a manifest file to deploy applications to Cloud Foundry is optional, it can help you standardize your application deployment, and it enables you to deploy multiple applications at once.

Procedure

1. Create a file called `manifest.yml` and add the contents using the following YAML conventions:
   - Begin with three dashes.
   - The `applications` block begins with a heading followed by a colon (:).
   - One dash and one space precedes the application `name`.
   - Additional lines are indented two spaces to align with the application `name`.

     Note: The manifest requires only an application name—additional properties are optional.

   The `memory` and `host` properties are optional.

   For example:

```
---
applications:
- name: <predix-app>
  memory: 512M
  host: <my_host>
```

2. Deploy your application to Cloud Foundry.

   The `cf push` command requires an application name regardless of whether or not you use a `manifest.yml` file to push your application. The `cf push` process looks for the `manifest.yml` file in the current working directory, or in the path provided on the command line using the `-f` option.

   If you provided the application name in the `manifest.yml` file and you are in the working directory for the application, all you have to do is enter: `cf push` on the command line.

   If the `manifest.yml` is not in the current working directory, provide the path to the `manifest.yml` file on the command line.

   For example:

   ```
   cf push -f ./some_directory/some_other_directory/
   alternate_manifest.yml
   ```

   If the application name is not provided in a `manifest.yml` file, you must provide it on the command line when you push your application.

   For example:

   ```
   cf push <my_application>
   ```

Setting Up Platform Services

Task Roadmap: Setting Up Platform Services

The following task roadmap shows the basic steps required to get started with Predix platform services.

#	Task	Description
1	Set up your trusted issuer.	Create an instance of the User Account and Authentication (UAA) service. UAA is the authorization server that each platform service uses for authentication. For more information, see Creating a UAA Service Instance on page 17. **Tip:** The maximum number of UAA instances that you can create in your space is 10. As a best practice, use the same UAA instance with each of your services.
2	Create an instance of your service.	Select the service that you need from the Predix.io catalog and create an instance of your service. For more information, see Creating a Platform Service Instance on page 19
3	Create OAuth2 clients to set up access to your service authenticated using UAA.	When you create a UAA instance, an admin client is automatically created for you to access UAA for additional configuration. You can create a new client for your service instance with specific scopes. If an Oauth2 client already exists, you can update the client to add your service instance. For more information, see Creating an OAuth2 Client on page 25.
4	Update the Oath2 client to add service specific scopes or authorities.	To enable your application to access a platform service, your JSON Web Token (JWT) must contain the scopes required for a platform service. For more information, see Updating the OAuth2 Client for Services on page 28.
5	Bind your application to the service instance.	To establish communication between your application and the platform service, you must bind the application to the service. For more information, see Connecting Your Application to a Platform Service Instance on page 21.
6	Start using your services.	Your service instances are listed on the **Console** page when you sign into Predix.io. For more information on using a specific service instance, use the **See Documentation** button on each service information page on Predix.io.

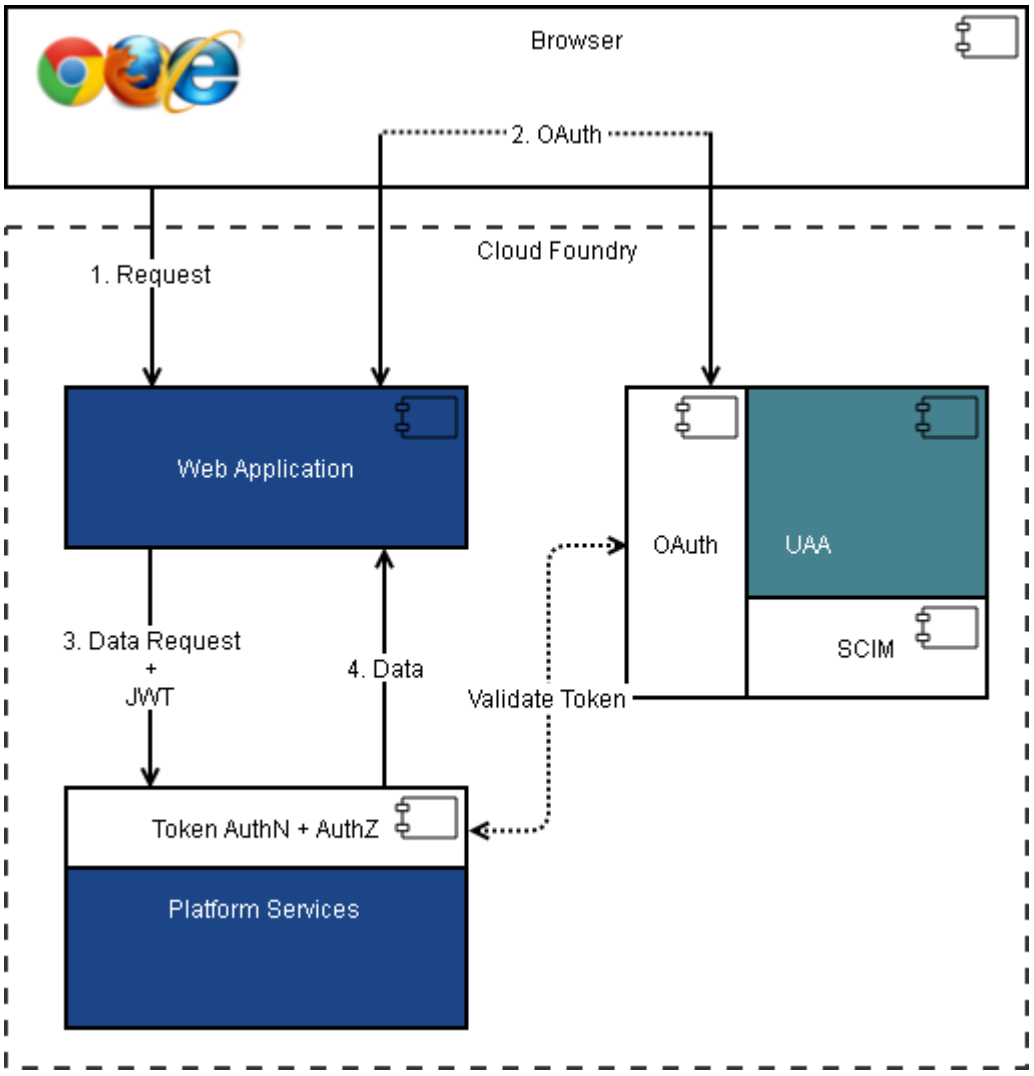
Understanding Security Setup of Platform Services

Predix platform services use OAuth2 for authorization. The services leverage the User Account and Authentication (UAA) web service to manage OAuth2 clients.

The User Account and Authentication (UAA) service is the primary authentication service on the Predix platform. It enables developers to add user authentication and authorization capabilities to their application. Application developers can obtain a UAA instance from the Predix marketplace and configure the instance to authenticate trusted users and clients for their application. UAA service offers a virtual OAuth2 server to customers to issue and validate tokens for client applications.

To use the platform services, you must first set up an instance of the UAA service as your trusted issuer. All access to the services is then authenticated using the designated trusted issuer.

The following diagram shows how platform services are integrated with UAA for authentication.



Creating a UAA Service Instance

You can create multiple instances of the UAA service in your space.

About This Task

As a best practice, first delete any older unused instances before creating a new one.

Procedure

1. Sign into your Predix account at https://www.predix.io.
2. Navigate to **Catalog** > **Services**, then click the **User Account and Authentication** tile.
3. Click **Subscribe** on the required plan.
4. Complete the fields on the **New Service Instance** page.

Field	Description
Org	Select your organization.
Space	Select the space for your application.
Service instance name	Enter a unique name for this UAA service instance.
Service plan	Select a plan.
Admin client secret	Enter a client secret (this is the admin password for this UAA instance). The client secret can be any alphanumeric string. **Note:** Record the client secret in a secure place for later use.
Subdomain	(Optional) Enter a subdomain you might need to use in addition to the domain created for UAA. You must not add special characters in the name of the subdomain. The value of sub-domain is case-insensitive.

5. Click **Create Service**.

Results

Your UAA instance is created with the following specifications:

- A client identifier (`admin`).

  Note: An `admin` client is required for bootstrap purposes. You can create additional clients to use with your application.
- A client secret (that you specified while creating the service).

To retrieve additional details of your instance, you can bind an application to your instance.

Using the Command Line to Create a UAA Service Instance

Optional procedure for using the command line instead of the graphical user interface to create a UAA service instance.

About This Task

You can create up to 10 instances of UAA service in your space. If you need additional instances, you must delete an older unused instance and create a new one.

Procedure

1. Use the Cloud Foundry CLI to log into Cloud Foundry.

   ```
   cf login -a <API_Endpoint>
   ```

   Note: If you are a GE employee, you must use the `cf login --sso` command to log into Cloud Foundry. After you enter your SSO, you will receive a one-time passcode URL. Copy this URL and paste it in a browser to retrieve your one-time passcode. Use this code with the `cf` command to complete the CF login process.

   Depending on your Predix.io registration, the value of `<API_Endpoint>` is one of the following:

   - Predix US-West
     `https://api.system.aws-usw02-pr.ice.predix.io`
   - Predix US-East
     `https://api.system.asv-pr.ice.predix.io`
   - Predix Europe
     `https://api.system.aws-eu-central-1-pr.ice.predix.io`

   For example,

   ```
   cf login -a https://api.system.aws-usw02-pr.ice.predix.io
   ```

2. List the services in the Cloud Foundry marketplace by entering the following command.

   ```
   cf marketplace
   ```

   The UAA service, `predix-uaa`, is listed as one of the available services.

3. Create a UAA instance by entering the following command.

   ```
   cf create-service predix-uaa <plan> <my_uaa_instance> -c
   '{"adminClientSecret":"<my_secret>","subdomain":"<my_subdomain>"}'
   ```

   where:

   - `cf` stands for the CLI command, `cloud foundry`
   - `cs` stands for the CLI command `create-service`
   - `<plan>` is the plan associated with a service. For example, you can use the `tiered` plan for the `predix-uaa` service.
   - `-c` option is used to specify following additional parameters.
     - `adminClientSecret` specifies the client secret.
     - `subdomain` specifies a sub-domain you might need to use in addition to the domain created for UAA. This is an optional parameter. You must not add special characters in the name of the sub-domain. The value of sub-domain is case insensitive.

   Note: Cloud Foundry CLI syntax can differ between Windows and Linux operating systems. See the Cloud Foundry help for the appropriate syntax for your operating system. For example, to see help for the `create service` command, run `cf cs`.

Results

Your UAA instance is created with the following specification:

- A client identifier (`admin`).

Note: An `admin` client is created for bootstrap purposes. You can create additional clients to use with your application.

- A client secret (that you specified while creating the service).

To retrieve additional details of your instance, you can bind an application to your instance.

Example

Create a predix-uaa service instance with client secret as admin and sub-domain as ge-digital:

```
cf cs predix-uaa tiered test-1 -c
'{"adminClientSecret":"admin","subdomain":"ge-digital"}'
```

This is how it appears in VCAP SERVICES when using the `cf env <app_name>` command:

```
"VCAP_SERVICES": {
"predix-uaa": [
   {
    "credentials": {
     "dashboardUrl": "https://uaa-dashboard.run.asv-
pr.ice.predix.io/#/login/04187eb1-
e0cf-4874-8218-9fb77a8b4ed9",
     "issuerId": "https://04187eb1-
e0cf-4874-8218-9fb77a8b4ed9.predix-uaa.run.asv-
pr.ice.predix.io/oauth/token",
     "subdomain": "04187eb1-e0cf-4874-8218-9fb77a8b4ed9",
     "uri": "https://04187eb1-
e0cf-4874-8218-9fb77a8b4ed9.predix-uaa.run.asv-
pr.ice.predix.io",
     "zone": {
      "http-header-name": "X-Identity-Zone-Id",
      "http-header-value": "04187eb1-
e0cf-4874-8218-9fb77a8b4ed9"
     }
    },
    "label": "predix-uaa",
    "name": "testuaa",
    "plan": "Tiered",
    "provider": null,
    "syslog_drain_url": null,
    "tags": [],
    "volume_mounts": []
   }
  ],
```

Creating a Platform Service Instance

You are required to set up a trusted issuer (UAA) before creating a platform service instance.

About This Task

After you have an instance of a trusted issuer in place, you can create an instance of a platform service for your application development.

Procedure

1. Sign into your Predix account at https://www.predix.io.
2. Navigate to **Catalog > Services** tab, and click the Service tile.
3. Click **Subscribe** on the required plan.
4. Complete the fields on the **New Service Instance**.

Field	Description
Org	Select your organization.
Space	Select the space for your application.
User Account & Authentication (UAA)	Select an existing UAA instance, or create a new one. For more information, see Creating a UAA Service Instance on page 17.
Service instance name	Enter a unique name for this service instance.
Service plan	Select a plan.

5. Click **Create Service**.

Creating a Platform Service Instance Using Cloud Foundry Commands

You can use the Cloud Foundry CLI to create a platform service instance.

Procedure

1. Use the Cloud Foundry CLI to log into Cloud Foundry.

```
cf login -a <API_Endpoint>
```

Note: If you are a GE employee, you must use the `cf login --sso` command to log into Cloud Foundry. After you enter your SSO, you will receive a one-time passcode URL. Copy this URL and paste it in a browser to retrieve your one-time passcode. Use this code with the `cf` command to complete the CF login process.

Depending on your Predix.io registration, the value of `<API_Endpoint>` is one of the following:

- Predix US-West
  `https://api.system.aws-usw02-pr.ice.predix.io`
- Predix US-East
  `https://api.system.asv-pr.ice.predix.io`
- Predix Europe
  `https://api.system.aws-eu-central-1-pr.ice.predix.io`

For example,

```
cf login -a https://api.system.aws-usw02-pr.ice.predix.io
```

2. List the services in the Cloud Foundry marketplace by entering the following command:

```
cf marketplace
```

3. Create a service instance by entering the following command:

```
cf create-service <service_name> <plan> <my_instance> -c
'{"trustedIssuerIds":["<uaa_instance1_issuerId>",
"<uaa_instance2_issuerID>"]}'
```

where:

- `<service_name>` is the name of the service in the Cloud Foundry marketplace. For example, the Asset service is `predix-asset`.
- `<plan>` is the plan associated with a service. For example, the `predix-acs` service can be associated with the `basic` plan.
- `<uaa_instance_issuerID>` is the `issuerID` of your trusted issuer (UAA instance), such as `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. You can retrieve this URL from the VCAP_SERVICES environment variable after binding your UAA instance to an application.

Note: Cloud Foundry CLI syntax can differ between Windows and Linux operating systems. See the Cloud Foundry help for the appropriate syntax for your operating system. For example, to see help for the `create service` command, run `cf cs`.

Cloud Foundry displays a message confirming that the service instance has been created. You can retrieve the URI of this instance from the VCAP_SERVICES environment variable after binding it to an application.

Connecting Your Application to a Platform Service Instance

To establish communication between the service instance that you have created and your application, you bind your application to the service instance.

Before You begin

Deploy your application to Cloud Foundry.

About This Task

When you bind your application to your service instance, Cloud Foundry provisions the service's connection details in the VCAP_SERVICES environment variable. Cloud Foundry runtime uses the VCAP_SERVICES environment variable to communicate with a deployed application about its environment.

You can retrieve the following instance details from VCAP_SERVICES environment variable:

- An `instance_uri` for your service instance.
- A `instance_GUID` is the `zoneID` for your service instance.
- HTTP header information to access your service instance. It includes:
  - `http-header-name` as `Predix-Zone-Id`
  - `http-header-value`
- An `oauth-scope` for your instance. The end-user token requires the scope to access the specific service instance.

If you bind to a UAA service instance, you get the following information in addition to `instance_uri` and `instance_GUID`:

- A `uaa_instance_issuerId` for your instance. The `issuerID` is required when you create an instance of another service that uses your UAA instance for authentication.

Procedure

1. Use the Cloud Foundry CLI to log into Cloud Foundry.

   ```
   cf login -a <API_Endpoint>
   ```

   Note: If you are a GE employee, you must use the `cf login --sso` command to log into Cloud Foundry. After you enter your SSO, you will receive a one-time passcode URL. Copy this URL and paste it in a browser to retrieve your one-time passcode. Use this code with the `cf` command to complete the CF login process.

   The value of `<API_Endpoint>` is available in the Predix welcome email that you receive when you register for a Predix.io user account. Depending on your Predix.io registration, the `<API_Endpoint>` value is one of the following:

   - Predix US-West
     `https://api.system.aws-usw02-pr.ice.predix.io`
   - Predix US-East
     `https://api.system.asv-pr.ice.predix.io`
   - Predix Europe
     `https://api.system.aws-eu-central-1-pr.ice.predix.io`

   For example,

   ```
   cf login -a https://api.system.aws-usw02-pr.ice.predix.io
   ```

   The command produces the following output:

   ```
   Email> <your_predix_login>
   Password> <your_predix_password>
   Authenticating...OK
   Targeted org <your_predix_org>
   Targeted space dev
   API endpoint: https://api.system.aws-usw02-pr.ice.predix.io (API
   version: <version>)
   User: <your_predix_login>
   Org:  <your_predix_org>
   Space: dev
   ```

2. Bind your application to the service instance by entering the following command:

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

   Your application is bound to the `<service_instance_name>` instance, and the following message is returned:

   ```
   Binding service <service_instance_name> to app <your_app_name> in
   org predix-platform / space predix as userx@ge.com...
   OK
   TIP: Use 'cf restage' to ensure your env variable changes take effect
   ```

3. Verify the binding by entering the following command:

   ```
   cf env <your_app_name>
   ```

Example

If you bind an application, myApp, to a UAA instance, the following messages are returned:

```
Getting env variables for app myApp in org predix-
platform / space security as userx@ge.com...
OK
...
  ],
  "predix-uaa": [
        {
        "credentials":
        {
         "issuerId":
          "https://ff27c315-d027-4d1d-
a30c-64f49b369ed9.predix-uaa.run.aws-usw02-
pr.ice.predix.io/oauth/token",
         "uri":
           "https://ff27c315-d027-4d1d-
a30c-64f49b369ed9.predix-uaa.run.aws-usw02-
pr.ice.predix.io",
          "zone": {
                    "http-header-name": "X-Identity-Zone-
Id",
                    "http-header-value": "ff27c315-
d027-4d1d-a30c-64f49b369ed9"
                   }
        },
        "label":
        "predix-uaa",
        "name":
        "my_uaa_instance",
        "plan":
        "free",
        "tags":
        []
        }
        ],
```

In this example, the following values are displayed:

- `uaa_instance_issuerId = https://ff27c315-d027-4d1d-a30c-64f49b369ed9.predix-uaa.run.aws-usw02-pr.ice.predix.io/oauth/token`
- `uaa_instance_uri = https://ff27c315-d027-4d1d-a30c-64f49b369ed9.predix-uaa.run.aws-usw02-pr.ice.predix.io`
- `uaa_instance_GUID = ff27c315-d027-4d1d-a30c-64f49b369ed9`

If you bind an application, myApp, to the ACS service instance, the following messages are returned:

```
Getting env variables for app myApp in org predix-
platform / space security as userx@ge.com...
OK
...
  ],
"predix-acs": [
```

```
   {
    "credentials": {
     "uri": "https://predix-acs.run.aws-usw02-
pr.ice.predix.io",
     "zone": {
      "http-header-name": "Predix-Zone-Id",
      "http-header-value":
"9615a95a-9275-4a82-926d-89f06cbe04e1",
      "oauth-scope": "predix-acs.zones.
9615a95a-9275-4a82-926d-89f06cbe04e1.user"
     }
    },
    "label": "predix-acs",
    "name": "acs-sample-instance",
    "plan": "Tiered",
    "provider": null,
    "syslog_drain_url": null,
    "tags": []
   }
  ],
```

In this example, the following values are displayed:

- `acs_instance_uri` = `https://predix-acs.run.aws-usw02-pr.ice.predix.io`
- `acs_instance_GUID` = `9615a95a-9275-4a82-926d-89f06cbe04e1`
- `http-header-name` = `Predix-Zone-Id`
- `http-header-value` = `9615a95a-9275-4a82-926d-89f06cbe04e1`
- `oauth-scope` = `predix-acs.zones.9615a95a-9275-4a82-926d-89f06cbe04e1.user`

Setting Secure Access to your Platform Service

Creating an OAuth2 Client

You can create OAuth2 clients with specific permissions for your application to work with Predix Platform services. Often this is the first step after creating an instance of a service.

About This Task

When you create an instance of UAA, the UAA Dashboard is available for configuring that instance of UAA. You can use the Client Management tab in the UAA Dashboard to create the OAuth2 clients.

If you are prefer using the UAA command-line interface (UAAC) instead of UAA Dashboard to create an OAuth2 client, see Using UAAC to Create an OAuth2 Client

Procedure

1. In the Predix.io Console view, select the Space where your services are located.
2. In the Services Instances page, select the UAA instance to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In UAA Dashboard, select the **Client Management** tab.

   The Client Management tab has two views, **Clients** and **Services** . The **Services** view displays the service instances that you have created for your services.

   Note: The service instances displayed in the Services view were created while using the UAA that you are trying to configure. Service instances that you created using other UAA instances are not displayed on this page.
6. Click **Create Client** to open the **Create Client** form.
7. Complete the **Create Client** form.

Field	Description
Client ID	Specify a name for the OAuth2 client you are creating.
Authorized Grant Types	Choose one or more of the following grant types: • **authorization_code** When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code. • **client_credentials** When you use the client credentials grant type, the OAuth2 endpoint in UAA accepts the client ID and client secret and provides Access Tokens. • **password** When you use the resource owner password credentials grant type, the OAuth2 endpoint in UAA accepts the username and password and provides Access Tokens. • **refresh_token** The refresh tokens are credentials used to obtain access tokens. You can choose this option to obtain refresh token from UAA. You can then use the refresh token to obtain a new access token from UAA when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope. • **implicit** When you use the implicit grant type, UAA directly issues an Access Token to the client without authenticating the client. This reduces the number of round trips required to obtain an access token. For more information on grant types, see RFC 6749.
Client Secret	Specify the password. It is important that you keep a note of this password. If lost, this password cannot be retrieved.
Confirm Client Secret	Reenter the client secret.

Field	Description
Redirect URI	Specify a redirect URI to redirect the client after login or logout (for example, `http://example-app.com/callback`). Use this URI when you start using UAA as the service provider for your external Identity provider. UAA uses the value of Redirect URI for `/oauth/authorize` and `/logout` endpoints. You must specify a Redirect URI value if you use the Authorization Code or Implicit authorization grant type. When you use the Authorization Code grant type, the Redirect URI is your application's endpoint or callback that expects user authorization code. When you use the Implicit grant type, the Redirect URI is the end point where UAA sends the bearer token. Unique Resource Identifier consists of: • Access Protocol, `http` or `https` • Domain or IP address • Access Port such as 80 or 443 • Path If you have a specific URL for your application callback, you can use that to set the Redirect URI value for the related client. For example, `https://your-app-domain.run.aws-usw02-pr.ice.predix.io/path1/path2/callback`. You can specify multiple values for Redirect URI as a list of allowed destinations that UAA server can redirect the users. For example, `https://yourappdomain1.run.aws-usw02-pr.ice.predix.io/path1/path2/callback`, `https://yourappdomain2.run.aws-usw02-pr.ice.predix.io/path1/path2/callback`. If the subdomain of your application is dynamic, you can set the value of Redirect URI using wilcards. For example, `https://*.your-app-domain.run.aws-usw02-pr.ice.predix.io/path1/path2/callback`. **Note:** You must only use '*' for a domain that is exclusive to your application (Such as `your-app-domain` in example above). This prevents the redirect to be routed to an application that you do not own. You cannot use `*` in the top domain and sub domain (such as `predix.io` in the example above).
Scopes	Scopes are permissions associated with an OAuth Client to determine user access to a resource through an application. The user permissions are for authorization grant types `authorization_code`, `password` and `implicit`. By default, the admin client is assigned all required scopes. For a new client, an administrator can select the scopes to be added based on client requirements. For a list of available scopes, see Scopes Authorized by the UAA. To use an OAuth2 client for your Predix Platform service instance, you must update your OAuth2 client to add scopes that are specific to each service after adding the client to the service instance.

Field	Description
Authorities	Authorities are permissions associated with the OAuth Client when an application or API is acting on its own behalf to access a resource with its own credentials, without user involvement. The permissions are for the `client_credentials` authorization grant type. By default, the admin client is assigned all required authorities. For a new client, an administrator can select the authorities to be added based on client requirements. The list of authorities matches the list of scopes. For a list of available UAA scopes, see Scopes Authorized by the UAA. To use an OAuth2 client for your Predix Platform service instance, you must update your OAuth2 client to add authorities that are specific to each service after adding the client to the service instance. **Note:** An admin client is not assigned the default authority to change the user password. To change the user password, you must add the `uaa.admin` authority to your admin client.
Auto Approved Scopes	Specify scopes that can be approved automatically for the client without explicit approval from a resource owner.
Allowed Providers	Specifies the names of the external identity providers, if any. This field is required if you are using external identity providers with UAA as a service provider.
Access Token Validity	Specifies the access-token expiration time in ms.
Refresh Token Validity	Specifies the refresh-token expiration time in ms.

Next Steps

Updating the OAuth2 Client for Services on page 28 for your service specific information.

Updating the OAuth2 Client for Services

To use an OAuth2 client for secure access to your Predix Platform service instance from your application, you must update your OAuth2 client to add additional authorities or scopes that are specific to each service.

About This Task

To enable your application to access a platform service, your JSON Web Token (JWT) must contain the scopes required for a platform service. For example, some of the scope required for Access Control service are `acs.policies.read acs.policies.write`.

The OAuth2 client uses an authorization grant to request an access token. Based on the type of authorization grant that you have used, you must update your OAuth2 client to generate the required JWT. For more information on how the OAuth2 client is created, see Creating OAuth2 client.

If you use the UAA Dashboard to create additional clients, the client is created for the default `client_credentials` grant type. Some required authorities and scopes are automatically added to the client. You must add additional authorities or scopes that are specific to each service.

In addition, the admin client is not assigned the default authority to change the user password. To change the user password, you must add the `uaa.admin` authority to your admin client.

Use the following procedure to update the OAuth2 client.

Procedure

1. In the Console view, select the Space where your services are located.

2. In the Services Instances page, select the UAA instance to configure.
3. Select the **Configure Service Instance** option.
4. In the UAA Dashboard login page, specify your admin client secret and click **Login**.
5. In UAA Dashboard, select the **Client Management** tab.

   The Client Management tab has two views, **Clients** and **Services**. The **Services** view displays the service instances that you have created for your services.

   Note: The service instances displayed in the **Services** view are the instances that you created using the UAA that you are trying to configure. The service instances that you created using some other UAA instance are not displayed on this page.
6. Select the **Switch to Services View** option.
7. In the **Services** view, select the service that you need to update.
8. Choose an existing client or choose the **Create a new client** option. If you chose to create a new client, follow the steps in Creating an OAuth2 Client on page 25.
9. Click **Submit**.
10. Click on the **Switch to Clients View** option.
11. In the **Clients** view, click the edit icon corresponding to the client added in the previous step.
12. Complete the **Edit Client** form.

Field	Description
Authorized Grant Types	Choose one or more of the following grant types: • **authorization_code** When you use the authorization code grant type, the client directs the resource owner to UAA, which in turn directs the resource owner back to the client with the authorization code. • **client_credentials** When you use the client credentials grant type, the OAuth2 endpoint in UAA accepts the client ID and client secret and provides Access Tokens. • **password** When you use the resource owner password credentials grant type, the OAuth2 endpoint in UAA accepts the username and password and provides Access Tokens. • **refresh_token** The refresh tokens are credentials used to obtain access tokens. You can choose this option to obtain refresh token from UAA. You can then use the refresh token to obtain a new access token from UAA when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope. • **implicit** When you use the implicit grant type, UAA directly issues an Access Token to the client without authenticating the client. This reduces the number of round trips required to obtain an access token. For more information on grant types, see RFC 6749.
Redirect URI	Specify a redirect URI to redirect the 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.
Scopes	By default, the client is assigned a few required scopes. For a new client, an administrator can select the scopes to be added based on the selected grant type. If you select the `authorization_code`, `password` and `implicit` grant type, you must update the scopes with service specific scopes. For a complete list of required scopes, see Authorities or Scopes Required for Platform Services on page 30. For a list of available UAA scopes, see Scopes Authorized by the UAA.

Field	Description
Authorities	By default, the client is assigned a few required authorities. For a new client, an administrator can select the authorities to be added based on the selected grant type. If you select the `client_credentials` grant type, you must update the authorities with service specific authorities. For a complete list of scopes to be added for each service, see Authorities or Scopes Required for Platform Services on page 30. For a list of available UAA authorities, see Scopes Authorized by the UAA.
Auto Approved Scopes	Specify scopes that can be approved automatically for the client without explicit approval from the resource owner.
Allowed Providers	Specify the names of the external identity providers, if any. This field is required if you are using external identity providers with UAA as a service provider.
Access Token Validity	Specifies the access token expiration time in ms.
Refresh Token Validity	Specifies the refresh token expiration time in ms.

Next Steps

You can complete the following additional tasks in UAA Dashboard:

- If you are using authorization grant type as Authorization Code, Implicit, or Resource Owner Password, you can manage users in UAA.
- You can create password policies for user passwords.
- You can set up external identity provider or use UAA as an identity provider. See Managing Identity Providers.

If you have completed your OAuth2 client setup, you can bind your application to your service instance.

Authorities or Scopes Required for Platform Services

When you create a new OAuth2 client, the client is assigned default scopes and authorities. You must add additional authorities or scopes that are specific to each service.

The following table lists the scopes and authorities specific to each platform service that you must add to your OAuth2 client.

Service Name	Authorities/Scopes
Access Control	• acs.policies.read • acs.policies.write • acs.attributes.read • acs.attributes.write • predix-acs.zones.<acs_instance_guid>.user This value is added by default if you use the UAA Dashboard. It is also generated in the VCAP_SERVICES environment variable as `oauth-scope` when you bind your application to your ACS service instance.
Analytics Catalog	analytics.zones.<service_instance_guid>.user (added by default)
Analytics Runtime	analytics.zones.<service_instance_guid>.user (added by default)
Asset	predix-asset.zones.<service_instance_guid>.user (added by default)
Blockchain as a Service	predix-blockchainapi.zones.<service_instance_guid>.user (added by default)

Service Name	Authorities/Scopes

Event Hub	• Publish
	  ◦ predix-event-hub.zones.<Predix-Zone-Id>.user
	  ◦ predix-event-hub.zones.<Predix-Zone-Id>.wss.publish
	  ◦ predix-event-hub.zones.<Predix-Zone-Id>.grpc.publish
	• Subscribe
	  ◦ predix-event-hub.zones.<Predix-Zone-Id>.user
	  ◦ predix-event-hub.zones.<Predix-Zone-Id>.grpc.subscribe

Tenant Management	• tms.tenant.read • tms.tenant.write • predix-tms.zones.<tms_instance_guid>.user (added by default)

Time Series	• Data ingestion
	  ◦ timeseries.zones.<Predix-Zone-Id>.user (added by default)
	  ◦ timeseries.zones.<Predix-Zone-Id>.ingest
	• Data queries
	  ◦ timeseries.zones.<Predix-Zone-Id>.user (added by default)
	  ◦ timeseries.zones.<Predix-Zone-Id>.query

View	• views.zones.<view_instanceId>.user (added by default) • views.admin.user • views.power.user

Additional Resources

Guides

These guides include tutorials, documentation, demos, code samples, templates, and more.

Explore the guides.

Videos

Learn development tips and techniques from Predix experts.

Watch the videos.

Resources

In addition to the guides and videos, additional resources include podcasts, a reference app, and more.

Check out more resources.

Community

The Predix community includes the newsletter, developer forum, blog, events, and more.

Explore the Predix community.

Support

See Predix support to file support tickets, view your open tickets, check the Predix platform status, and find information about common errors and FAQs.