# Progress Spark Toolkit Migration Guide

Migration instructions for existing PMFO projects

August 2019

# Summary

The **Progress Spark Toolkit** is a reference implementation of the Common Component Specification (CCS) framework, and replaces a portion of the code previously obtained as part of the Progress Modernization Framewok for OpenEdge (PMFO). In the past this code was implemented as part of the Professional Services organization of Progress, but is now available as an open-source offering. As such, it comes as-is with no warranty and differs slightly from the bundled code previously implemented. The following guide will highlight the differences to allow you to migrate seamlessly to the new repository.

# Prerequisites

To provide the best experience with the **Progress Application Server (PAS)** and **Progress Developer Studio (PDSOE)** it is recommended that you be on the latest service pack of OpenEdge. The source provided with Spark should be compatible with both OpenEdge 11.6 and 11.7, with the latter being preferable due to significant simplifications of security options and improved support for Single Sign-On and OAuth. Use of a **64-bit Windows** installation and **OE 11.7.4 or later** is required, and at least the Progress Developer Studio for OE component present for development purposes.

Access to the repository is provided through **Git** and you may benefit from having a suitable Git client installed on your workstation. As a suggestion, **Git for Windows** and **TortoiseGit** will provide a seamless integration with Window Explorer. Some automated tasks will be performed using **Ant** which will already be present in your DLC directory if using 11.7 or later.

# Obtaining Code

Previously, all code was obtained through an SVN repository hosted by the PS organization. The latest open-source code is now available through GitHub via distinct repositories which have split the previous codebase by distinct audience and purpose:

**Spark-Toolkit-Demos** – Demo projects, quick-start template, and support files

<https://github.com/progress/Spark-Toolkit-Demos>

**Spark-Toolkit** – Original server-side framework code and procedure libraries

<https://github.com/progress/Spark-Toolkit>

# Code Changes

Some obsolete and unused code has been removed from the codebase, and a few external files have been renamed (eg. the ABLUnit base class PMFOUnit.cls is now SparkUnit.cls). Though the most significant change between the PMFO-based codebase and the new open-source project is the name of the Procedure Library. Originally the server-side code was packaged within a “**PMFO.pl**” file, and is now built as a “**Spark.pl**” library. The reason for this change is to separate the CCS-based reference implementation (Spark) from the name of the overall modernization approach (PMFO). Comments and documentation should now refer to all artifacts as part of the “Progress Spark Toolkit” or similar. All other directory structures and class packages should have remained unchanged.

# Code Migration Process

Please follow the steps below to migrate to the new framework distribution. This process assumes that you are working with the source code within a PDSOE project and have already deployed/published to a PAS instance.

1. Retrieve the **Spark-Toolkit** code from the GitHub repository.
2. Locate and copy the **Spark.pl** library in the **/dist/** folder.
3. Delete any existing **PMFO.pl** file and replacing it with the **Spark.pl** file
   1. For the project, this is expected to be within the /AppServer/ directory.
   2. For your PAS instance, this is the CATALINA\_BASE/openedge/ directory.
4. Update your PROPATH entries to use the new **Spark.pl** file.
   1. Right-click on your PDSOE project and restart the Progress AVM.
   2. Adjust the /**conf/openedge.properties** file in your PAS instance.
5. Recompile your project, publish the changes, and restart your PAS instance.

You should also update the **/AppServer/merge.openedge.properties** with the change in PROPATH to the new Spark.pl file. This will ensure the modified library name is not forgotten if you have to rebuild your PAS instance elsewhere.

# Service Migration Process

A major change as of the v4.4.0 release is the deprecation of the Spark version of a “Data Object Handler” class. This class, while named the same, was a gateway into the RouteManager class to execute code dynamically. However, this pattern did not utilize the same code and processes as used by the out-of-the-box **DataObjectHandler** class. To rectify this issue and consolidate all dynamic code execution into a tested and supported class, we need to switch use of the Spark-supplied class with the product-supplied class. Since the WebHandler class cannot be changed in an existing ABL Service, we will need to create a new service with the proper class and relative URI endpoint(s). The process below will walk you through the creation of a new service which utilizes the appropriate OpenEdge class.

1. Locate a “**DataObjectService**” in the **Defined Services** area of your project.
2. Double-click the service to view the existing properties.
   1. If the existing WebHandler class is configured as **Spark.Core.Handler.DataObjectHandler** continue with the steps below.
   2. Note the full list of **Resource URI’s** for reference later (default “**/pdo**”).
3. Close the properties panel and delete the existing **DataObjectService** service.
4. Create a new **ABL Service** with the name of “DataObjectService” just as before.
5. Use the **WEB** transport, but not the option for “Create a Data Object Service”.
6. For the **WebHandler** class use “OpenEdge.Web.DataObject.DataObjectHandler”
7. For the Resource URI there are two paths which can be followed:
   1. Using only the Spark Entity pattern.
   2. Mixing PDO and Spark Entity patterns.
8. Follow the paths below for completing your new service…

Only Spark Entity Pattern

If you wish to maintain only use of the Spark Entity pattern (which uses the CatalogManager to discover available resources) then you may continue to use the “/pdo” endpoint.

1. From the property pane, set a new **Resource URI** which is just **/pdo**
   1. If more than 1 Resource URI was used previously, add those now.
2. Click on the Finish button to save your changes and close the dialog.
3. Be sure to add the new ABL Service to your PASOE server.

Mixing PDO and Spark Entity Patterns

If you wish to potentially mix both the standard PDO pattern with annotated business entities along with the Spark Entity pattern, then you must avoid any collision of URI’s as handled by the DataObjectHandler class.

1. From the property pane, set a new **Resource URI**, for instance **/api**
   1. If more than 1 Resource URI was used previously, add those now.
   2. Change any use of “**pdo**” with “**api**” as necessary.
2. Click on the **Finish** button to save your changes to the new ABL Service.
3. Make sure the new ABL Service is added to your PASOE server.
4. With the change to the default URI, we need to make adjustments elsewhere.
   1. Within your client-side code which used the **/api** prefix should instead of **/pdo** when loading the Data Service Catalog (typically found in **login.html** and **app.html**).
5. Update to the latest **catview.html** and **devinfo.html** pages in /static/
   1. These can be found in the **Spark-Toolkit-Demos** project.
6. Adjust any existing BE resources to use the adjusted **serviceURI** property.
7. Adjust the **oeablSecurity.csv** to use the new URI prefix of **/api** where applicable.

# Configuration File Changes

In addition to the changes to the service, there were multiple adjustments made to the configuration files which drive the operation of the manager classes. Note that sample configs are also included in the **/cfg/** folder for further reference.

1. Adjust the **catalog.json** config file to use the new URI prefix for services.
   1. The URI prefix should match the one chosen in the previous steps.
   2. Also add a “**ServicePrefix**” property to the “**General**” object.
   3. Note the added “**CatalogService**” property and initial value.

"General": [{

"ApiVersion": 4.6,

"BusinessRoot": "Business",

**"CatalogService": "Spark.Core.Service.ICatalog",**

"IdProperty": "id",

"PreLoader": "",

"ReadFilter": "filter",

**"ServicePrefix": "api",**

"EnableDebugs": false

}]

1. Adjust the **services.json** config file to include the following **Service Mappings**:

{

"Service": "Spark.Core.Manager.IClientContext",

"Implementation": "Sports.Spark.Core.Manager.ClientContext"

}, {

"Service": "Spark.Core.Service.ICatalog",

"Implementation": "Spark.Core.Service.Catalog"

}, {

"Service": "OpenEdge.Web.DataObject.IServiceRegistry",

"Implementation": "OpenEdge.Web.DataObject.ServiceRegistryImpl"

}, {

"Service": "Progress.Web.IWebHandler",

"Implementation": "OpenEdge.Web.DataObject.DataObjectHandler"

}

1. Adjust the **services.json** config file to include the following **Service Life Cycles**:

{

"ServiceMatch": "Spark.Core.Service.Catalog",

"LifeCycle": "Session"

}, {

"ServiceMatch": "OpenEdge.Web.DataObject.ServiceRegistryImpl",

"LifeCycle": "Session"

}, {

"ServiceMatch": "OpenEdge.Web.DataObject.ServiceRegistryLoader",

"LifeCycle": "Session"

}

1. Adjust the **startup.json** config file to remove objects referring to the following:
   1. Spark.Core.Manager.MessageManager
   2. Spark.Core.Manager.RouteManager
   3. Spark.Core.Manager.StatsManager
2. Recompile your project, publish any changes, and restart your PAS instance.

**NOTE:** The config files **handler.json** and **message.json** may be removed entirely. For these, their manager classes which utilized them have been removed.

# WebApp Changes

If upgrading from 11.7.3 to 11.7.4 it is advised that you update the oemanager WebApp for your deployed PASOE instances, especially in development. You can use the **Tomcat Manager** to undeploy and deploy the WebApp as located in the **DLC/servers/pasoe/extras** folder as **oemanager.war**. This will provide you with the necessary features for viewing/running the **OpenAPI** endpoints for monitoring your PASOE instance, as well as reading other OpenAPI data such as via the new Catalog service to be created in the next section.

After deploying the latest oemanager WebApp, locate the **oemgrSecurity-container.xml** file under the **webapps/oemanager/WEB-INF** folder of your PASOE instance. Uncomment the line which begins with **<intercept-url pattern="/doc/\*\*"** to allow access to the /docs endpoint. Restart the PASOE instance and simply visit the **http://<hostname>:<port>/oemanager/** endpoint to be redirected to the OpenAPI (Swagger) document viewer for the **PASOE Management API’s**.

**NOTE:** You may be prompted to log in with credentials for this page, which by default are **tomcat/tomcat** (the “container” security model uses that of the Tomcat server itself).

# 

# Adding Support for OpenAPI

This feature assumes that you are utilizing the **DataObjectHandler** for your available API resources, and have previously configured the standard Spark/startup.r and Spark/shutdown.r as your sessionStartupProc and sessionShutdownProc entries, respectively.

1. In PDSOE, create a new **ABL Service** called “CatalogService” or “catalog”.
2. Use the **WEB** transport, but not the option for “Create a Data Object Service”.
3. For the **WebHandler** class use “OpenEdge.Web.DataObject.DataObjectHandler”
4. Add the following Resource URI’s:
   1. /catalog
   2. /catalog/openapi
5. Click the **Finish** button to save changes and close the dialog.
6. Open the **oeablSecurity.csv** file for your project and add the following:

"/web/catalog","GET","permitAll()"

"/web/catalog/openapi","GET","permitAll()"

1. Save and close the security file.
2. Be sure to add the new ABL Service to your PASOE server and publish changes.
3. Restart your PASOE server to pick up the modifications in security.

# Mapping the Service

We also require a mapping file to describe our new service endpoints. To do this, create a “**catalog.map**” file at PASOEContent/WEB-INF/openedge with the following JSON:

{

"services": {

"catalog": {

"version": "1.0.0",

"operations": {

"/": {

"GET": {

"contentType": "application/json",

"statusCode": 200,

"entity": {

"name": "Spark.Core.Service.Catalog",

"function": "getCatalog",

"typeof": "Progress.Lang.Object",

"arg": [{

"ablName": "catalog",

"ablType": "class Progress.Json.ObjectModel.JsonObject",

"ioMode": "Output",

"msgElem": {

"type": "BODY",

"name": "catalog",

"ioMode": "Output"

}

}]

}

}

},

"/openapi": {

"GET": {

"contentType": "application/json",

"statusCode": 200,

"entity": {

"name": "Spark.Core.Service.Catalog",

"function": "getOpenApiCatalog",

"typeof": "Progress.Lang.Object",

"arg": [{

"ablName": "catalog",

"ablType": "class Progress.Json.ObjectModel.JsonObject",

"ioMode": "Output",

"msgElem": {

"type": "BODY",

"name": "catalog",

"ioMode": "Output"

}

}]

}

}

}

}

}

}

}

# Testing the Catalog Output

Once the mapping file has been published to your instance, we can then confirm the changes to the server instance worked. For this we need only to confirm the presence of output from our new API endpoints. To begin, enter the following URL in your browser after replacing with your host/port as necessary for your PASOE instance:

**http://<hostname>:<port>/sports/web/catalog**

The resulting GET of this URL should produce the same output as expected by the JSDO (meaning this is the description for Progress Data Object Services). If you wish to confirm the OpenAPI output, simply change the URL to the following for testing:

**http://<hostname>:<port>/sports/web/catalog/openapi**

If a request of this URL produces output in JSON format then we are ready to test the viewer display of the OpenAPI data.

# Viewing the OpenAPI Output

To provide a user-friendly view of the generated output we can reuse the oemanager WebApp as configured previously. To do this we need only pass a new catalog endpoint.

1. Enter the following URL with the host and port for your PASOE instance:

**<hostname>:<port>/oemanager/doc/api-docs?url=/sports/web/catalog/openapi**

1. You should now see the OpenAPI documentation for your application.
2. Now to test some requests to your server via the viewer:
   1. Change the **Server** option to use **{ctx}/web{basePath}**
   2. Set the **ctx** value to **/sports**
   3. Set the **basePath** to **/api**
      1. Use **/pdo** if this was how you configured the **DataObjectService**
   4. Attempt to run one of the endpoints available for your application. Note that if a login is required there is no means of performing this step with the current viewer. You will need to log in against the same server via the **/auth/login.html** page to set the necessary cookie.