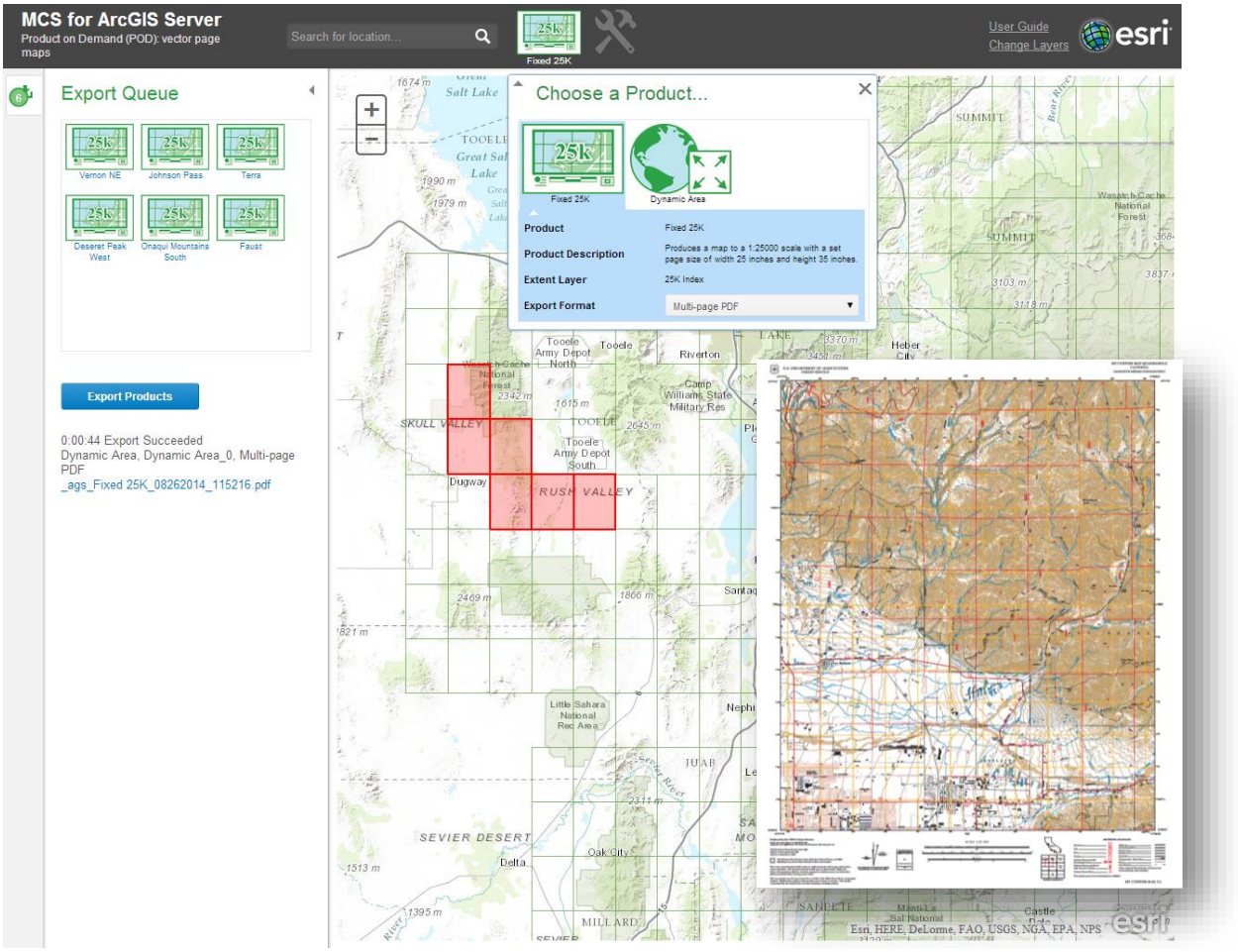
Product on Demand (POD) Deployment Guide

Table of Contents

Introduction to Product on Demand (POD)

Product on Demand (POD) makes it easy to discover, generate and share map products with high cartographic detail over the web. The POD system, or framework, includes a web application hosted on a web server that interacts with ArcGIS for Server to produce high-quality maps in several formats such as PDF, MAP PACKAGE, and JPEG. POD can be configured to produce maps at various scales with custom specifications while using real-time data.



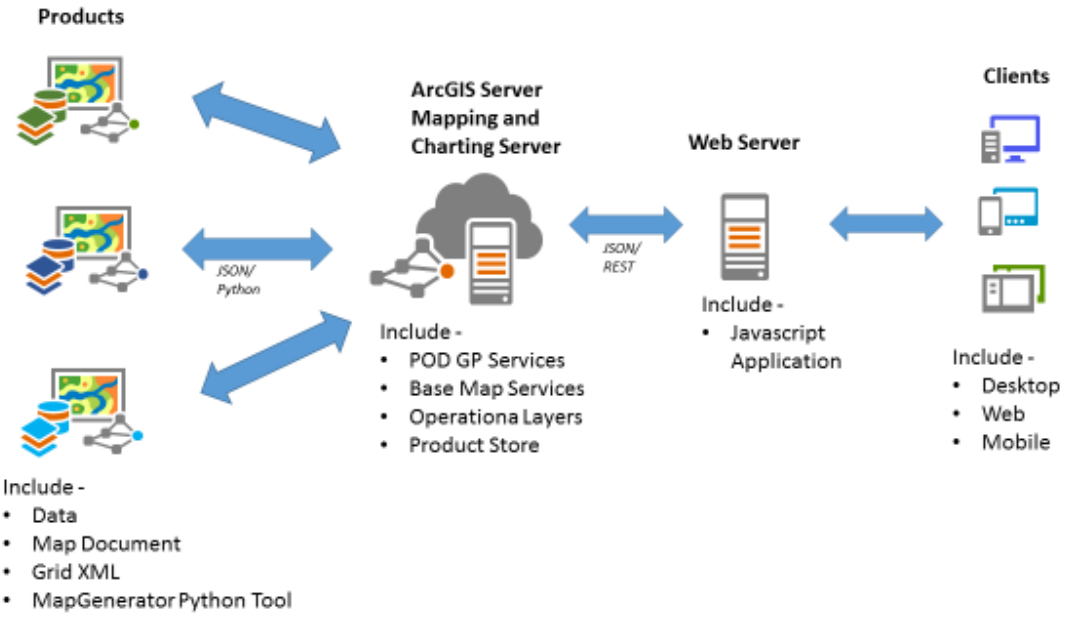
Key Features

- Provides centralized location for production of a variety of map products using the web application
- Provides easy-to-use interface for discovery of geographic locations to generate maps
- Allows selection of predefined mapping extents using map services
- Provides tools to generate custom map size based on scale, area, or paper size
- Provides functionality to generate multipage PDFs
- Provides preview capability for quick snapshot of final product

- Utilizes advanced cartographic capabilities such as masking, symbol level drawing, grids and graticules, and dynamic map content to generate high-end map output

POD Architecture

The following diagram represents the architectural components of a POD system.



This document provides guidance for setting up the different components of POD. It is recommended that the user of this document has an understanding of Publishing Services using ArcGIS for Server and working knowledge of Webservers and JavaScript is beneficial.

Prerequisites for POD Deployment

Web Server

The POD application is compatible with web servers such as Internet Information Services (IIS) and Apache Tomcat, WebSphere and WebLogic. Contact your server administrator for information about setting up a web server.

ArcGIS 10.3 for Server

Learn more about installing ArcGIS for Server

Production Mapping 10.3 for Server

Learn more about Mapping and Charting solutions for Server

ArcGIS Web Adaptor (Optional)

If you are preparing your site for a production deployment, install ArcGIS Web Adaptor. The web adaptor provides a link between your enterprise web server and your ArcGIS for Server site.

For more information, see About ArcGIS Web Adaptor.

Note: If ArcGIS Web Adaptor is being used with IIS7.5 or greater, then modify the response HTTP Response header for the ArcGIS Web Adaptor to "Accept – Ranges". This setting helps eliminate any download issues for maps created through POD. For IIS follow the steps below:

1. Open IIS Manager.
2. In the Connections pane, expand Sites.
3. Expand the website where the Web Adaptor is deployed and select the Web Adaptor application.
4. Double-click HTTP Response Headers feature.
5. Right-click and add a new header called "Accept-Ranges" and set its value to "none".

Download POD files from GitHub

To set up POD, access the files you will need from the POD Esri GitHub repository found here. Follow these steps to download the repository:

1. Go to the releases tab
2. Download the latest release
3. Extract the downloaded release zip to a local folder on your machine.

Now that the necessary POD folder structure is created and files extracted, you can proceed with setting up POD.

Note: You will need admistrative permission to perform some of the steps in the Deployment Guide. Programmers that choose to customize POD can choose to fork or clone the POD repo. For more information on how to work with Esri Github repositories, click here.

Configure the Web Server for POD

The web application used in POD is a JavaScript-based frontend user interface to generate high-quality printable maps. The files and folders required for JavaScript application are located in the WebServer folder at the extracted location. The table below describes the nature of contents in the extracted WebServer folder:

Sub Folder/ Files Level 1	**Sub Folder/ Files Level 2**	**Description**
css	POD.css	Styling information for the web application.
images	ProductTypes	Add product images to this folder to use as product logo.
	Various image files	Images referenced in the web application, including Application logo.
js	nls	Resource bundles for internationalization.
	podconfig.js	Main configuration file that is modified by a technical administrator to configure POD.
	Various .js files	Application logic to build the website dynamically based on configurations defined in the podconfig.js file.
proxy	proxy.ashx	Application logic for the proxy.
	proxy.config	Server URLs to execute transactions between the web application and ArcGIS for Server.
index.html		The default page for the web application.
UserGuide.html		The container HTML page that displays the User Guide.
UserGuide.pdf		An instructional PDF file that describes how to use a particular POD site.

To configure the web server, copy all the contents of the MCS_POD\WebServer folder to the web root directory of your web server.
Note: In Microsoft IIS 7, the default web server directory is <your IIS installation folder>\inetpub\wwwroot\. If you are using a different version, modify accordingly.

Next we will continue with setting up ArcGIS for Server for POD.

Configure ArcGIS for Server for POD

POD is configured on ArcGIS for Server by moving the POD application files to the location of your ArcGIS for Server directories and updating your ArcGIS for Server site with POD services. This can be done on a new or existing ArcGIS for Server site. Contact your system administrator to configure an ArcGIS Server site if missing or see Create a new site.

Follow these steps to configure your ArcGIS for Server site for POD:
1 **Enable sharing on the arcgisserver directory.** By default, this location is <ArcGIS for Server installation drive>\arcgisserver.

2 **Create a folder named 'MCS_POD' in the ArcGIS Server directory**. The folder MUST be named MCS_POD for the system to work correctly. By default, this location is <ArcGIS Server installation drive>\arcgisserver.

3 **Place the ArcGIS Server contents from the extracted location folder created above.** Copy all the files and folders from the sub directory ArcGISServer in the extracted location to <ArcGIS Server installation drive>\arcgisserver\MCS_POD. These are the necessary files to configure ArcGIS for Server for POD. Below is description of the contents.

Sub Folder Level 1	**Sub Folder/ Files Level 2**	**Description**
Products		Location to group products and related files. New Product are defined in this folder.
	Fixed 25K	Sample of a Fixed type product. Refer to the instructions in this folder to download product files for Fixed 25K from GitHub
	Dynamic Area	Sample of an Area type product
	Dynamic Page	Sample of a Page type product
	Dynamic Scale	Sample of a Scale type product
	Starter Product	Contains the basic files that can be used as starting point for building new products.
Service DefinitionFiles		Service definition files that are used to publish ArcGIS for Server services deployed with POD.
	Calculators.sd	Geoprocessing service used for calculating map properties such as area, scale, and size on paper based on user input
	Gateway.sd	Geoprocessing service for relaying data from the JavaScript front-end to the map generation Python tools.
MapserviceData		Map document and Database used in the Sample map service
Tools		Python toolbox script used for the geoprocessing service definition
Utilities	Utilities.py	A python configuration file on ArcGIS for Server that manages output location and utility functions for integration between JavaScript and Python

4 **Register 'MCS_POD' folder with your ArcGIS Server DataStore.** If default locations were used, this folder will be at <ArcGIS Server installation drive>\arcgisserver\MCS_POD. This location will also be used in the following steps to copy and configure different products and their related data files. Therefore, registering it as a Data Store will provide ArcGIS Server with the necessary access to all product files needed to generate a finished map. Learn more about registering a folder with ArcGIS Data Store.

5 **Update Utilities.py**: If defaults were accepted, this file will be located in < ArcGIS Server installation drive>\arcgisserver\MCS_POD\Utilities. This file contains variables which store the location of three paths that are used at various points in the application. It is recommended that UNC paths be used for the following variables to ensure that all required directories can be accessed.

5.1 ***shared_products_path****:* This variable stores a path to the Products folder where different products and their respective files are located. If all the contents were copied correctly, the location of this folder will be at <ArcGIS Server installation drive>\arcgisserver\MCS_POD\Products. Update the value assigned to the shared_products_path variable to point to \\<ArcGIS Server machine name>\arcgisserver\MCS_POD\Products.

5.2 ***output_directory****:* The application returns any output to the ArcGIS Server output directory. If ArcGIS Server was installed with default settings, then this location will be at <ArcGIS Server installation drive>\arcgisserver\directories\arcgisoutput. Update the value assigned to the output_directory variable to point to \\<ArcGIS Server machine name> \arcgisserver\directories\arcgisoutput. This path can also be accessed as a virtual path, or URL.

5.3 ***output_url:*** This is a URL representation of the output_directory referenced in the earlier step. Since all the ArcGIS Server directories are virtual, it allows web applications to request outputs created in these locations via a URL. The URL is http://gisserver.domain.com:6080/arcgis/rest/directories/<directory name>. Update the value assigned to output_url variable to point to http://<gisserver.domain.com:6080>/arcgis/rest/directories/arcgisoutput.

To learn more about ArcGIS Server directories and virtual paths see About Server Directories.

6 **Publish POD ArcGIS Server services using service definition files.** If defaults were accepted, these files will be located in \\<ArcGIS Server>\arcgisserver\MCS_POD\ServiceDefinitionFiles. Using your ArcGIS for Server site manager, publish the provided geoprocessing services.
To learn more about using service definition files to publish ArcGIS Server services, see Publishing a service definition to server in Manager.

6.1 ***Publish the Calculators Service:*** This service contains an ArcGIS geoprocessing service used for calculating map properties such as area, scale, and size on paper based on user input. The service has four tasks. Using your ArcGIS for Server site manager, publish the Calculator.sd file with all default settings.

6.2 ***Publish the Gateway Service:*** This service contains a geoprocessing service definition for managing the interaction between the JavaScript frontend and the map generation Python tools. Using your ArcGIS Server site manager, publish the Gateway.sd file with all default settings.
Note: Change the default and set a greater time out for the Gateway service based on the performance of your ArcGIS Server to ensure successful publishing of the maps. (e.g., 60000 seconds). Consider updating the maximum file age property on the output directory of your ArcGIS Server site to ensure that the map file is returned and not deleted before the process completes. Learn more about tuning your geoprocessing service.

To ensure that all services are published correctly, verify their REST endpoints in your services directory http://<arcgisserver.domain.com>:6080/arcgis/rest/services/

Service	**Example REST URL**
CalculateExtent	…Calculators/GPServer/CalculateExtent
CalculateScale	…Calculators/GPServer/CalculateScale
CalculatePageSize	…Calculators/GPServer/CalculatePageSize
CalculateStripMap	…Calculators/GPServer/CalculateStripMap
Gateway	…Gateway/GPServer

7 **Integrate with web server:**

7.1 ***Update the podconfig.js:*** The podconfig.js file on the WebServer contains the REST URLs for POD geoprocessing services. If defaults were accepted, the file will be at <your IIS install folder>\Inetpub\wwwroot\<application folder>\js\podconfig.js. Edit this file in a text editor and modify the values assigned to gpCalculateExtentUrl, gpCalculateScaleUrl, gpCalculatePageSizeUrl, gpCalculateStripMapUrl and gpCalculateScaleUrl properties with the respective REST URLs for the services published earlier.

7.2. ***Update the proxy with Gateway REST URL:*** A proxy is used to execute transactions between the web application and ArcGIS Server. Edit the proxy.config file inside the proxy folder by adding a server URL entry for Gateway service. If defaults were used this will be at <your IIS install folder>\Inetpub\wwwroot\<application folder>\proxy.
Example: <serverUrl url="http://<yourServer>/arcgis/rest/services /Gateway/GPServer/" matchAll="true"></serverUrl>

This completes the deployment for POD on ArcGIS Server and the WebServer. To create and define new products, continue to the next section or explore setting up the samples referenced in the appendix.

Add a New Product

A Product represents a map that can be created using POD. It is characterized by its content, layout and mapping context for which that map can be used. It serves as a template for creating several maps with the same layout but covering different areas on ground. Multiple products can be hosted on the POD website. The process for authoring a new product requires ArcGIS for Desktop and Production Mapping extension.

The following steps outline the process used to define new product for POD:

1 **Create a new folder in the MCS_POD\Products folder on ArcGIS Server**. If the defaults were accepted, this will be in <ArcGIS Server installation drive>\arcgisserver\MCS_POD\Products. Provide a name that is unique and identifies the product. This name will appear in the web application and is used again in the configuration file.

2 **Prepare the necessary components that will be used for Map Creation.** The map creation process includes files that contain the specifications used for generating a high-quality cartographic map. Below is a list of the minimum components that should be placed in the folder created in the earlier step. Additional files required for map creation that are specific to your business logic may also be identified and moved to this location.

> 2.1 ***Map Document****:* This represents the general layout of the page from which the map will be created. It contains the necessary data, surround elements, and text elements that should appear on the final map. Learn more about creating and managing layouts. For POD, it is recommended that the page size of the Map Document is defined in "inches".
> **Note**: If the data in your map points to a file geodatabase, it is recommended that the map document be stored with fixed paths not relative paths.
>
> 2.2 ***Grid Template XML****:* This is used to calculate the ground area, map scale, or map page size used for the map. It can also be used to generate grid and graticules layers for the final map. Learn more about using a grid to determine an area on the ground based on scale and map size.
>
> 2.3 ***Python toolbox file****:* This is a Python tool that will be used to process the data and create a high-quality map that will be used for final publishing. This tool can implement any Python module, including those that are native, third-party, or available with ArcGIS for Server and Mapping and Charting for Server. Learn more about Python Toolbox. For the Python toolbox to work with POD, it must implement a recommended structure. Below is a general outline of this structure.
>
> 2.3.1 Import required modules.
> 2.3.2 Define module level constants.
> 2.3.3 Add the paths to the shared location in the __init__() method inside the tools class.

2.3.4 Define the getParameterInfo method with exactly two parameters – one input and one output.
2.3.5 Implement the tools execute method with code to unpack the web applications input; use any of the product files; pan, zoom, scale the map; set the correct page; and perform an export. Custom logic that involves your business process can be injected into before the export. Refer to the POD_MapGenerator.pyt in the Starter Product for a sample of the minimum structure for creating a product python toolbox file. If defaults were used then this location will be at <ArcGIS Server installation drive>\arcgisserver\MCS_POD\Products\Starter Product
2.3.6 Save the toolbox file in your Products folder.
2.3.7 Note down the name of the pyt file and the tool name that performs this logic.

Note: As you test your tool with the application you may need to update the python tool. Restart the Gateway Service, to test any changes made to the python tool.

3 **Identify if the product type for your map product**. The type can be Fixed, Area, Scale or PageSize. There are four product types supported with POD, depending on the map properties that are known and those that the application must calculate given a set of fixed or user-defined values. The following table lists each product type based on its parameters. Use this table to determine which one fits your needs the best. More discussion on product types is available in the Appendix.

Type	Area on Ground	Map Scale	Page Size
Fixed	Fixed	Fixed	Fixed
Area	Application will calculate	User-defined	User-defined
Scale	User defined	Application will calculate	User-defined
PageSize	User-defined	User-defined	Application will calculate

4 **Publish an Extent Layer Map Service**. This is required for a Fixed type product and can be optional for the other types. In the case of a Fixed type product, the map service represents predetermined areas on the ground for map creation. In case of the remaining dynamic products, a map service is used to display an area on the ground that is valid for mapping. Lear more about publishing mapservices here.

5 **Add an entry for this index layer in the ExtentLayers property in the podconfig.js file.** If defaults were used, this file will be at <your IIS install folder>\Inetpub\wwwroot\<application folder>\js\podconfig.js. This step ties the published service with the web application. The property must be named ExtentLayers.

Syntax:

```
ExtentLayers : [
                    {value: "100K Index",
                     url: "http://myarcgisserver/arcgis/rest/services/mcs_pod/100K/MapServer",
                     sublayer: 0,
                     data0: "MapSheetID" }
                    ]
```

Where:
value = A unique string that represents the name of the extent layer. This will be used to reference the layer when defining a product.
url = The REST URL for the published extent map service.
sublayer = The index of the sublayer when a map service contains multiple layers.
data0 = Field name that will be used to derive attribute information for the selected feature. .
Additional data fields can be defined by adding incremental data# property, for example data0: "MapSheetID" and data1: "MapName." These data fields are primarily used when defining fixed type products and are useful (a) to display additional metadata for the map in the web application, (b) participate in a search query when the search feature of the web application is used, and (c) packaged as a parameter in the data passed to the server so it can be further consumed by the python tool to perform map automation
Note: Extent layers are required for Fixed type products. These act as a selectable feature service in the web application where users can select from predefined areas to generate a map. Other product types may also use Extent layer. In these cases, the extent layer will only serve as a display but not for any of the other uses listed above.

6 **Identify Page Margin should be applied**: A page margin can be configured to apply padding between the main map frame and the page. The values provided as margins are subtracted from the page dimension when the application performs any of its calculations. The page margin can be used to accommodate marginalia such as map title, legend, and North arrow. This is typically used for dynamic products where page size will change based on user input.

Identify the margin values in the order of top, right, bottom, left units. The following units are valid: inches, centimeters, millimeters, points, or percentage.

Example:
1 1 1 1 INCHES indicates that a 1-inch margin should be applied to any page size for area calculation. So for a page size of 8 x 11 inches, a map area covering 6 x 9 inches on the page will generate leaving a margin of 1 inch on each side of the map.

7 **Add the product configuration to the podconfig.js file**. Once all the product components are identified, these are coded into a podconfig.js file. If defaults were used, this file will be at <your IIS install folder>\Inetpub\wwwroot\<application folder>\js\.

7.1 **Define a new product attribute table property with a list of the minimum required attributes**. A new product will appear in the Choose a Product dropdown. To do this define a new property in the podconfig object and assign its value to a list of required attributes for this product. These attributes are identifiers used to dynamically update the web application for that product and are packaged and sent to the python tool on arcgis server for creating the final map.

Example:

```
MyProductAttrTable: [
                      //list of product attributes
                    ]
```

The minimum set of attributes are different for the different product types. See attributes defined in the podconfig.js file for property Fixed25KTable, DynamicAreaTable, DynamicScaleTable, DynamicPageTable as examples of Fixed, Area, Scale and PageSize type products respectively, More information on how to define products attributes is explained in the upcoming section on Customizing POD. The following table describe the meaning of each attribute if its required to be defined in the Product attributes:

Attribute	**Meaning**	**Required by**
type	Product type identified earlier	All types
productName	Name of the product. This should be the same name that was given to the folder in Step 1. This text will also appear in the web application for the user to select.	All types
description	User-friendly text describing the product. This text will appear in the web application as a tooltip for that product.	All types
makeMapScript	Product Map Python toolbox created earlier	All types
toolName	Product Map Map Python tool created earlier	All types
imageOverride	Property set to generate previews in export queue	All types
extentLayer	Assigned to the extent layer in Step 5. This is required for Fixed type products.	All types
basemapLayer	Assigned to a basemap that should draw when the product is selected	All types
mapExtent	Used to configure custom extents for the product. See “Customizing Product Extents” in the next section for more information. Set value to null for zooming to full extent of extent layer.	All types
mxd	Product MXD identified in Step 2.1	All types
gridXml	Product grid XML identified in Step 2.2	All types

pageMargin	If padding should be applied, provide valid margin values as determined in Step 6 in the order top, right, bottom, left units. An example for this can be 1 1 1 1 INCHES.	All types
exporter	Assigned to an export format that should be used to create the final map.	Optional
scale	Assigned to a scale value that should be used to create the final map	Area, PageSize
pageSize	Assigned to a page value that should be used to create the final map	All types
orientation	Assigned to a page orientation value that should be used to create the final map	Area, Scale
mapCommandDefault	Default tool that should be selected when a user selects a product	All types
mapCommandButtons	List of tools that should appear in the toolbox for that product. This references the mapCommands list.	Optional
mapCommandContextMenu	List of tools that should appear in the right click menu when this product is selected. This references the mapCommands list.	Optional
roundToNearest	Property to set when any calculations done by the application should be rounded. See the section on Customize POD for more info.	Optional

Example of a product attribute list:

```
MyProductAttrTable: [
    { attr: "type", value: "Fixed" },
    { attr: "productName", value: "Fixed 25K" },
    { attr: "description", value: "Produces a map with POD" },
    { attr: "makeMapScript", value: "Fixed25K_MapGenerator.pyt" },
    { attr: "toolName", value: "MapGenerator" },
    { attr: "basemapLayer", value: "Topographic" },
    { attr: "extentLayer", value: "25K Index" },
    { attr: "mapExtent", value: null },
    { attr: "layersOverride", value: true },
    { attr: "mxd", value: "CTM25KTemplate.mxd" },
    { attr: "scale", domain: "ScaleList", value: 250000, passToServer: true, isEditable: false },
    { attr: "pageSize", displayName: "Page Size", value: "A3", isEditable: false },
    { attr: "orientation", displayName: "", value: "Portrait" },
    { attr: "gridXml", value: "CTM_UTM_WGS84_grid.xml" },
    { attr: "pageMargin", value: "4.5 8 23 8 CENTIMETERS" },
    { attr: "imageOverride", value: true },
    { attr: "thumbnail", value: "Fixed 25K.png" },
```

```
      { attr: "exporter", value: "PDF"},
      { attr: "mapCommandDefault", value: "select_point" },
      { attr: "mapCommandButtons", filter: ["select_point", "select_polyline", "-",
"extent_point", "extent_move"] },
      { attr: "mapCommandContextMenu", filter: ["zoomin", "zoomout", "pan", "-",
"layer_zoomto", "-", "clear_selected", "clear_all"] },
      { attr: "roundToNearest", value: false }
   ],
```

7.2 **Add an "attrTable" entry for the newly added product to ProductDefinitons property in the podconfig.js file.**

Syntax**:**

```
ProductDefinitions: [
                              { attrTable: "MyProductAttrTable"} ]
```

Where,
"attrTable" = Property Name for product property defined in step 7.1

8 **Restart the published Gateway service**. Using your ArcGIS Server site manager, navigate to the services tab. Locate the Gateway service published in the earlier section. Stop and restart the service so the new changes are initialized in ArcGIS Server.

This will create a new product to be used with the POD web application.

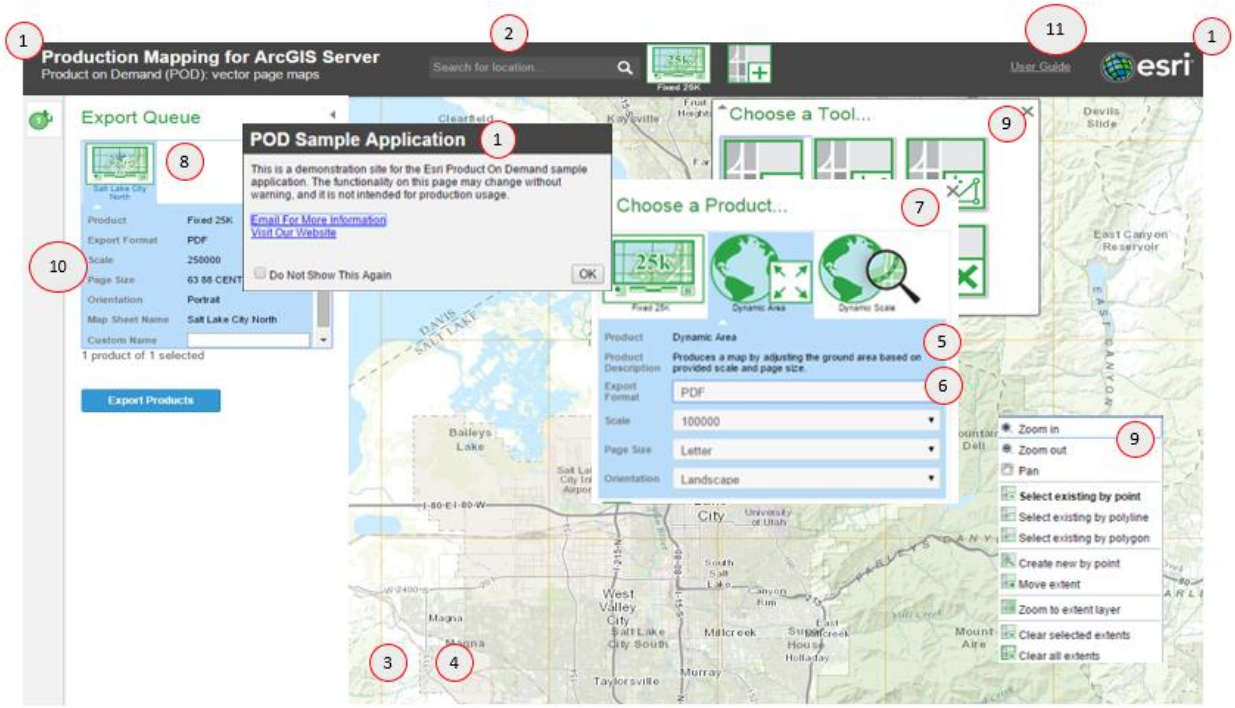
Additional features within the POD application can be customized if needed. The following section provides additional information and scenarios for advanced customization in POD.

Customize POD

The POD web application can be customized to meet an organization's specific needs. The majority of the customization is done by editing the podconfig.js. If defaults are accepted, this file will be located in <your IIS install folder>\Inetpub\wwwroot\<application folder>\js\podconfig.js.

Customization of the interface is possible for the following:

1 Application Splash Screen, Text and Logo
2 Application Services - Geocoding Service and Geometry Service
3 Basemaps and Extent Layers
4 Customizing Product Load Extents
5 Lists available with Application – Exporter, Scale, PageSize
6 Adding more lists and text control for use in the application
7 Configuring List of Products
8 Configuring the Export Queue
9 Customizing the Tool List for Products
10 Rounding Calculation
11 User Guide



1 Application Splash Screen, Title and Logo

The POD web splash screen appears the first time the site is loaded. It provides the user with context and an introduction to the web application. It can be configured to contain custom text, a web URL, and an email as contact info for your organization. The splash screen initiates if enabled in your configuration when the site loads. A user can disable it by checking the Do Not Show Again checkbox. The tiles and logo appear in the top header of the web application.

1.1 **Splash Screen:** To customize the content of the splash screen, modify the following properties of AppLevelSettings the podconfig.js file.

Syntax:

```
// WELCOME SCREEN SETTNGS
AppLevelSettings : {
        isSplash: true,
        isSplashContactInfo: true,
        splashTitle: "POD Sample Application",
        splashText: "This is a demonstration site for the Esri Product On Demand
        sample application. The functionality on this page may change without
        warning, and it is not intended for production use.",
        splashEmailDesc: “Email for more information”,
        splashEmail: "email@domain.com",
        splashEmailAlias: "CONTACT ALIAS",
        splashWebsiteDesc: “Visit our Website”,
        splashWebsite: http://www.<yourwebsite>.com
        splashDoNotShow: “Do not show this again”
        },
```

Where:

isSplash: Provide *true* if this splash screen should pop up on site load; provide *false* to suppress it
isSplashContactInfo: Provide *true* if a link to email should be included
splashTitle: Provide text that is displayed as the title of the splash screen dialog box
splashText: Provide custom text that will be displayed in the splash screen dialog box
splashEmailDesc: Provide text that can be used as an email hyperlink
splashEmail: Provide an email address as contact info
splashEmailAlias: Provide an alias for the email
splashWebsiteDesc: Provide text that can be used as your website name hyperlink
splashWebsite: Provide a web address to your organizational website home page.
splashDoNotShow: Text that should appear for the Do Not Show check box label

1.2 **Title and Logo:** The title and logo components used in the web application can be modified by editing the podconfig.js. file. Prior to doing this, ensure that the image that will be used as the logo is copied to the *images* folder located in <your IIS install

folder>\Inetpub\wwwroot\<application folder>. Edit the podconfig.js file in a text editor. Modify the values assigned to the following properties in AppLevelSettings:

applicationTitle : Update with text to use as the title
applicationSubtitle : Update with text to use as the subtitle
logoImage : Update with the name of the logo image file (e.g. "images\mylogo.png")

2 Application Services

The POD application uses arcgis services for processing requests. When installed, the default application settings point to a live services hosted by Esri, but this can be changed to a service that is local to your network for better performance or security reasons.

2.1 **The Geocoding Service** is used by the web application to find match results when the user types in the search tool. Set the new service URL to the geocodeServiceUrl property of AppLevelSettings in the podconfig.js file.

2.2 **The Geometry Service** is used by the web application to perfom calculations that allow it to render the ground area accurately. Set the new service URL to the geometryServiceUrl property of AppLevelSettings in the podconfig.js file.

2.3 **Other services** that are used by the application include the Geoprocessing services which is discussed in the section on Configure ArcGIS for Server for POD and map services, which is discussed next.

3 BaseMap and Extent Layers

Basemaps and operational extent layers are drawn in the web application when the application is loaded or when a user changes product. The basemaps and extent layers are referenced in separate lists in the podconfig.js and can be used across multiple products.

3.1 **BaseMapLayers:** This is a list of REST URLs for basemaps that will be available for use within the web application. A basemap can also be set as the default basemap when site first loads.

Add a new basemap layer by modifying the *BasemapLayer* property in the podconfig.js file.

Syntax:

```
BasemapLayers :
                [
                { value: "Topographic", url:
    "http://server.arcgisonline.com/ArcGIS/rest/services/World_Topo_Map/MapServer"},
                 { value: "Imagery",
        url:"http://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServ
        er" } ],
```

Where:
value = The text used to reference the basemap when defining a product.
url = The URL for the basemap service; this could be any of the basemaps available on ArcGIS online, or on your organisation's site.

Tip: To mash up different map services to use as a basemap, append the URLs to individual map services in a comma-separated list and use this as the string for the URL definitions above.

Example: { value: "Imagery with Labels", url: "http://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer,http://server.arcgisonline.com/ArcGIS/rest/services/Reference/World_Boundaries_and_Places/MapServer" },

To set an application default basemap, update the value to defaultBasemapLayer property in AppLevelSettings and assign the value to the basemap you want to as the default.

Syntax:

```
AppLevelSettings: {
                    // other app level settings
                   defaultBasemapLayer: "Light Gray Canvas"},
```

To add a basemap reference for a product, add the basemapLayer attribute and set its value to the value of the basemap you want to use.

Example:

```
MyProductAttrTable: [
            { attr: "type", value: "Fixed" },
            { attr: "productName", value: "My 50K Map" },
             //append to other required attributes
             { attr: "basemapLayer", domain: “Basemaplayers”, value: "MyBaseMap1" }
              ]
```

Note: “attr” must have value “basemapLayer” for Base maps to work correctly.

3.2 **ExtentLayers:** An extent layer is another layer that is drawn in addition to the basemap. It can be used to add more data to the web map. In the case of Fixed type products this is required and serves as a selectable layer for creating Fixed products. Extent Layers, are defined in the ExtentLayer property in the podconfig.js file and can be referenced in Product. For more discussion on this, please see section “Add a New Product”.

4. Customizing Product Load Extents

When working with the POD web application, you will need to interact with the map to locate and create mapping areas to produce a map product. You can customize the initial extent of the web map so users are quickly guided to the specific areas where a map product is valid. For instance, an organization that specializes in creating maps in South East Asia can configure POD such that the map automatically pans and zooms to the extent of South East Asia.
Note: A free online utility can be used to determine the extent coordinates. See here

There are three ways to apply a custom extent:

4.1 **For the Entire Application:** This setting is applied when the application is first launches in the browser. During this process, the extent of the web map area matches the settings defined in the application. This is valid for all products staged in your POD application.

To apply this setting, modify the podconfig.js file.

Search for the property named BaseProductTable and update the values for the mapExtent attribute to match the xmin, ymin, xmax, ymax coordinates of the region you want the application to pan to when launched.

Syntax:

```
BaseProductTable : [ // this table defines base product properties for all products
                     //other base properties
                     { attr: "mapExtent", value: { xmin: -15204166.06, ymin: 1593253.26,
                      xmax: -6398620.40, ymax: 7463617.03,
                      spatialReference: { wkid: 102100 } } },
                      ],
```

4.2 **For a Products Extent Layer**: This setting is applied when a product is selected from the Product List and the user clicks the Zoom To Full Extent button. During this process, the extent of the web map zooms to the full extent of an extent map service layer referenced for that product. An extent layer for a product is a map service that works like an operational layer and is associated with a product. For example, if you have an operational layer that represents the area covering North America, you can use this layer as your Product extent layer and define the map extent to use its bounds.

To apply this setting, you must first add the map service that represents your extent layer to a product definition. See Section 4 – “Creating a New Product” for more information. Once defined, modify the podconfig.js file and update the values for the mapExtent attribute to ‘null’. This ensures that the full extent of your extent layer service will be used.

Syntax:

```
Fixed25KTable : [ // properties specific to the Fixed25K Product
        { attr: "extentLayer", value: "25K Index" },
        { attr: "mapExtent", value: null } // use value: null for zooming to fullExtent
        ];
```

4.3 **Customizing a Products Extent Layer:** Like the product extent layer, this setting is also applied when a product is selected and the user chooses to zoom to full extent. However, during this process, instead of zooming to the full extent of the map service, a subregion within that service can be specified. This allows you to utilize the same map service for different products but have different areas to represent different mapping regions. For instance, a 1:25000 scale product may only be valid around major urban centers, so coordinates for the city bounds can be provided. However, a 1:50000 scale product may be valid for a larger area, perhaps at the state level, so coordinates for the state bounds can be provided.

Similar to product extent, custom product extent also requires that an extent layer is defined for that product. You can modify the podconfig.js file and update the values of the mapExtent attribute in a specific product table to provide the xmin, ymin, xmax, and ymax for a subregion within that extent layer.

Syntax:

```
Fixed25KTable : [
        // properties specific to the Fixed25K Product
        { attr: "extentLayer", value: "25K Index" },
        { attr: "mapExtent", value: { xmin: -12645741.95, ymin: 4399685.75,
         xmax: 12107625.28, ymax: 5187292.89, spatialReference: { wkid:
        102100 } } }
        ],
```

5 Customizing the Lists available with the application

Many lists are present in the podconfig file and can be used with the different products. Primarily these include the Exporters, Page Sizes and Scale Lists.When used a drop down is presented to the user with all the available values from the lists so the user can make a choice when creating a map. For example, a user can choose to create a map in PDF format if “PDF” is defined in the list of exporters. These lists can be modified with by adding or deleting items. A list is referenced by a product when its is defined in the Products attribute. You can also filter a product to use a subset of a list defined in the application. See below on how to modify these list definitions and refernce them in a Product Defintion.

Note: Because these are high level properties, it is recommended that items be filtered rather than removed from the list. See some examples below.

5.1 **Exporters:** The supported output formats that can be used for the final map. The application supports the following publishing and ArcGIS packaging formats out-of-the-box:

PDF, JPEG, TIFF, Map Package, production PDF, and layout GeoTIFF. To add a new format, edit the podconfig.js file in a text editor, then add an item to the Exporters property. Syntax:

```
Exporters : [
            {value: “ExporterName”, extensions : “FileExtension”}
            ]
```

Where:
value = This value used to reference default exporters for a product. It also appears in the exporters list for users to choose.
extensions = A valid file extension for the exporter type

Note: In order for a new export format to work with the application, you will need to modify the Utilities.py file located on your ArcGIS Server machine. If defaults were accepted, this file will be located at < ArcGIS Server installation drive>\arcgisserver\MCS_POD\Utilities. Modify the method def export_map_document() and append an elif code block to handle the new export format.

To use a list in a Product, add the exporter attribute to the Product attribute table and define a filter with a subset of exporters values.
Example:

```
MyProductAttrTabel : [
                    { attr: "type", value: "Fixed" },
                    { attr: "productName", value: "My 50K Map" },

                    //append to required attributes
                   { attr: "exporter", domain: “Exporters”, filter: [“PDF”,"MPK"] }
                    ]
```

To set a default exporter for a product, add the “exporter” attribute to the product attribute table, then use the value to set a default.
Example:

```
MyProductAttrTable : [
                    { attr: "type", value: "Fixed" },
                    { attr: "productName", value: "My 50K Map" },
                    //append to required attributes
                    { attr: "exporter", domain: “Exporters”, value: "MPK", filter: [“PDF”,"MPK"]}
                    ]
```

5.2 **ScaleList:** This is a list of the scale values available for use within the web application. This list can be modified by editing the ScaleList property.

Syntax:

```
ScaleList :  [
```

```
{ value: 5000, displayName: "1:5,000" }
]
```

Where:
value = The value passed to the ArcGIS Server for map creation. This should be in integer format. This is used to reference the default scale when defining a product.
displayName = This will also appear in a choice list when user is interacting with the application.

The scale list appears in a drop-down list for dynamic products. To define a default scale for a product, reference a “scale” attribute in the Products attribute table.

Example:

```
MyProductAttrTable : [
                {attr: “type”, value: “Fixed”}
                { attr: “productName”, value: “My 50K Map”
                //append to required attributes
                { attr : “scale”, domain: “ScaleList”, value: 5000}
                ]
```

5.3 **PageSize List:** This is a list of page size values available for use within the application. This list can be modified by editing the podconfig.js file in a text editor and adding items to the PageSizeList property. The application supports the following page sizes out-of-the-box – A0, A1, A2, A3, A4, A5, ANSI C, ANSI D, ANSI E, Letter, and Legal. Additionally, page sizes can be defined as custom page sizes.

Syntax:

```
PageSizeList : [
                { value: "A0", displayName: "A0", tooltip: "A0 (33.1in x 46.8in)" },
                { value: "32 44 CENTIMETERS", displayName: "Custom Size 32*44 cm",
tooltip: "Custom Size 32*44 cm" },
                ]
```

Where:
value = The value passed to the backend for map creation. For supported PageSizes, the value must be the same as the name. This is used to reference page size when defining a product. For a custom page size, thed value must be define as WIDTH HEIGHT UNITS.
displayName = This will also appear in a choice list when a user is interacting with the application.
tooltip = This will display as tooltip text when a user hovers over a page size. It can contain additional information such as dimensions of the page.

To define a default page size for a product, add a “pageSize” attribute in the Product.

Example:
MyProductAttrTable : [
{attr: “type”, value: “Fixed”}
{ attr: “productName”, value: “My 50K Map”}
//append to required attributes
{ attr : “pageSize”, domain: “PageSizeList”, value: “A4”}
]

6. Adding more controls for use in the application

Earlier we took a look at modifying existing lists provided with the application. In this section we will review the process for creating new lists and text controls. This provides capabilities for extending the products parameters so additional values can be used for map creation. For example, if you need the ablity to allow users to choose to a specific map edition in order to generate a map, this can be done by providing a choice list with all the possible map editions for that product. There are two ways to extend a product’s parameters.

6.1 **Add a choice list:** A choice list is a drop-down list of predefined values. To configure a choice list, first we define a list of choices and then reference them in a product definition. Similar to other lists discussed previously, these lists should be a collection of {value:””, displayName:””} pairs.
Examples:
MyCustomChoiceList : [
{ value:”MapEdition1”, displayName:”1st Edition”},
{ value:”MapEdition2”, displayName:”2nd Edition”},
{ value:”MapEdition3”, displayName:”3rd Edition”},
{ value:”MapEdition6”, displayName:”4th Edition”}
]
Where,
value = The value that will be packaged in the list of parameters id passToServer is true
displayName = This is what will appear in the user interface if this list is consumed in a product

To consume a choice list in a product so that it appears in the Product options or the export queue, it should be referenced in the product’s attribute table or instance table.
Example:
MyProductAttrTable : [
{attr:”type”, value: “Fixed”}
{attr:”productName”, value: “My 50K Map”}
//append to required parameters
{attr:”mycustomattr”, domain:”MyCustomChoiceList”,

value:" MapEdition1", displayName: "Choose Edition",
isEditable: true, passToServer: true}
]

Where:
attr = A name given to the custom attribute for its product definiton. This will also be the parameter name that is passed to the server for any backend processing when passToServer is true.
domain = The dictionary variable that contains a list of possible values for this choice list.
value = Default value that should appear when the control is initialized.
displayName = Text that will appear in the web application as a label for the choice list. If displayName is set to "" then the control is hidden from the user interface.
isEditable = Indicates if the the user can interact with it. True will allow the user to choose from the list. False will disable interaction in the web application.
passToServer = Indicates if an attribute and its values should be passed as a parameter to the backend server for processing.

Products can reference all items, or a subset of items, from a list. To reference a subset of items, add a "filter" when adding a choice list to the specific product.
Example:
{attr:"mycustomattr", domain:"MyCustomChoiceList", filter:["MapEdition1", "MapEdition2"], value:"MapEdition1", displayName: "Choose Map Edition", isEditable: true, passToServer: true}

6.2 **Add a text box**: A text box will allow the user to input their own values which can also be used in map creation. For instance, a user can provide a custom name that appears as the title of the map.

To add a text box control, simply add a new attribute to the product table or instance table that defines the text box and its value.

Example:
MyProductAttrTable : [
{attr:"type", value: "Fixed"}
{attr:"productName", value: "My 50K Map"}
//append to required parameters
{attr:"mycustomnameattr", value:"",
displayName: "Custom Map Name", isEditable: true, passToServer: true}
]

Where:
attr = A name given to the custom attribute for its product definition. This will also be the parameter name that is passed to the server if passToServer is true.
value = A default value that should appear when the control is initialized.
displayName = Text that appears in the web application next to the text box.
isEditable = Indicates if the user can interact with it. True will allow the user to type in the box. False will disable interaction in the web application.
passToServer = Indicates if an attribute and its values should be passed as a parameter to the backend server for processing.

7. Configuring a List of Products

New products are added to POD by gathering the Product related files and making updates to the podconfig.js file. The process for defining a new product is described in detail in an earlier section on —"Add a new Product". Products in POD appear under the Choose a Product button where user can choose a product and set its configuration to generate maps. Below is summary of some key points to remember when creating a new product to add to POD.

- The properties for each product are grouped together in a list defined as the products attribute table.
  Example:
  MyProductAttrTable : [
  // required attributes and additional attributes
  ]
- Required attributes must be defined in each product table, this is defined in the section on "Add a new product".
- It must have an "attr" property which serves as an identifier.
- It must have a "value" property which defines a value. The values can setup so they can be modified by the user when interacting in the web application.
- An attribute can have a "passToServer" property. When set to true, the value of the attribute will be packaged and sent to server.
- An attribute can have a "displayName" property. When set to a non-empty string, the attribute will be displayed to the user in the "Choose a Product" dialog in the web application.
- An attribute can have an "isEditable" property. When set to true, user can change the value of the attribute by selecting option from a dropdown or entering values in a text box in the web application. Refer to the section on "Adding more controls for use in the application" for details.
- If the possible values on an attribute depend on a predefined list, then a "domain" property can be added. A subset of domain can be defined by setting a "filter" property.
- An attribute maybe hidden from the user interface but always packaged as a parameter in the data passed to the server to perform map automation. This is done

by setting the attributes value, displayName to “” and passToServer to True.
Example:
{attr : “exporter”,displayName : ””, domain : “Exporters”, value: “PDF”, passToServer: true}

- All products inherit their attributes from BaseProductTable. To override attributes of the BaseProductTable, use of the same attr name and redefine its properties. If new attributes are used across multiple products it can be defined in the BaseProductTable property. Because the BaseProductTable is a high level property, it is recommended that attributes be added but not removed.
- A product appears in the POD web application only when its attribute table is referenced in the ProductDefinitions property of podconfig.
  Example:
  ProductDefinitions : [
  { attrTable: "MyProductAttributeTable"}
  ]

8. Configuring the Export Queue

The export queue in the web application is the tab on the left side that gets populated with product maps that are ready for export when selected in the web application. It serves as a stack tray where users can go to review maps — look at the previews, add custom names, reorder the stack before printing maps. Certain customizations are possible for the export queue.

8.1 **The number of maps in the export queue**: You can control the number of maps that can be added to the export queue and the number of maps that can be packaged at one time to send to the server for generating maps. This is done by modifying podconfig.js and changing the values assigned to maxProductInExportGrid and maxProductToExport properties in AppLevelSettings.

Example:

```
AppLevelSetting : {
        //in addition to other app level settings
        maxProductInExportGrid: 50,
        maxProductsToExport: 5
        }
```

8.2 **Generation of Preview images for maps in the export queue**: To change when a preview for a map should be generated, modify that products attribute table and add an imageOverride property.
Example:

```
MyProductAttributeTable: [
        //in addition to other required properties
        {attr: “imageOverride”, value : true}
        ]
```

Note: The make map script must implement the preview logic for this to work correctly. See the steps in “Add a new Product” for more info.

8.3 **Changing the map details in the export queue**: When a map for a product is added to the export queue, certain information is displayed including Product name, Map Name etc. The details displayed here can be modified by creating and/ or modifying a product’s instance table. Similar to the product attribute table, the product instance table is a collection of attributes for values that should appear in the export queue for a selected map.
Example:
MyProductInstanceTable : [// instance attributes and additional attributes]
The rules for defining instance attributes is similar to the rulese that apply for a product table. Below is a summary of the key point:

- Each attribute must have an “attr” property which serves as an identifier.
- Each attribute must have a “value” property.
- An attribute can have a “displayName” property. When set to a non-empty string, the attribute will be displayed to the user in the “Export” queue mapsheet details.
- An attribute can have an “isEditable” property. When set to true, user can change the value of the attribute by selecting option from a dropdown or entering values in a text box in the web application. Refer to the section on “Adding more controls for use in the application” for details
- An instance attribute that should be packaged as a parameter in the data passed to the server to perform map automation should be defined in that Products attribute list and its passToServer set to true.
- All products instance tables inherit their attributes from BaseProductTable, BaseInstanceTable and its specific Product table. If new attributes are used across multiple instance tables it can defined in the BaseInstanceTable property. Because this is a high level property it is recommended attributes be added but not removed.
- This details for a product’s map sheet appears in the export queue only when the instance table is referenced in the ProductDefinitionsProperty along with the product attribute table.
  ProductDefinitions : [
  { attrTable: "MyProductAttrTable", instanceTable: “MyProductInstanceTable”}
  ]

9. Customizing the Tool List

There are several tools available for interacting with the web map in the web application. The tools available for creating and managing products can be customized for each product. The tools are accessible through the Toolbox button and the context menu. These two access points can be customized independently of one other. To customize tools that are present in the toolbox and context menu for a product add the ‘mapCommandsButtons’ and ‘mapCommandContextMenu’ attribute to the Products attribute table, then add a filter with the tools that should be added respectively.

Example:
MyProductAttrTable : [
{attr:”type”, value: “Fixed”}
{attr:”productName”, value: “My 50K Map”}
//append to required parameters
{ attr: "mapCommandButtons", filter: ["extent_point", "extent_polyline", "extent_move",”-“,” select_point”,” select_polyline”] },
{ attr: "mapCommandContextMenu", filter: ["zoomin", "zoomout", "pan", "-", "clear_selected", "clear_all"] }
]

Below is a list of all the tools in the MapCommands variable in the podconfig.js file:

Tool Name	Use
Zoomin	Allows you to zoom in on the map
zoomout	Allows you to zoom out on the map
Pan	Allows you to pan the map
select_point	Allows you to select an extent by clicking on the map
extent_point	Allows you to create a custom extent by clicking on the map (Typically used for the Dynamic Area Product)
select_polyline	Allows selection of many extents by drawing a line on the map
extent_polyline	Allows you to create a series of custom extents by drawing a line on the map (Typically used for the Dynamic Area Product)
select_polygon	Allows you to select extents by drawing a polygon on the map
extent_polygon	Allows you to create a custom extent by drawing a polygon on the map (Typically used for Dynamic PageSize and Dynamic Scale Product)
extent_move	Allows you to move custom extents on the map
layer_zoomto	Allows you to zoom to the product load extent
clear_selected	Allows you to clear a selected extent on the map
clear_all	Allows you to clear all the extents on the map
-	Allows you to add a separator between tools for better usability

10. Rounding calculations

For few of the product types in particular, notably the Scale and Page Size type of products, the resulting values for the scale and page size are calculated by the calculator services. In most cases the results of the calculations will not be rounded, leading to outputs such as 1:502345 scale or 7.345 inches for page width. These results can be rounded by adding a ‘roundToNearest’ attribute to the product variable. This allows you to use more standards outputs for the map creation. There are two ways to apply rounding:

10.1 **Rounding to a nearest value from a List**: This restricts the rounded value to always round to a value within a list. This rounding applies to both scale and page size type products. For scale

product, the resulting output will always round to next smaller scale value in the Scale list. Similarly, for page size product, the resulting output will always round to the next larger page size value in the Page Size list. Do this to set the roundToNearest value as true for a product:

Example:
MyProductAttrTable : [
{attr: “type”, value: “Scale”}
{ attr: “productName”, value: “My Scale Map”}
//append to required attributes
{ attr : “roundToNearest”, value: true}
]

10.2 **Rounding to the nearest value:** This allows rounding to use the nearest Nth value for the resulting output and applies only to the Scale and Area type product. To do this, set the roundToNearest value to the Nth value that you want to use for rounding. For example, if you want results to be rounded to the nearest 1000, follow the example below.
Example:
MyProductAttrTable : [
{attr: “type”, value: “Scale”}
{ attr: “productName”, value: “My Scale Map”}
//append to required attributes
{ attr : “roundToNearest”, value: 1000}
]

11. User Guide

The web application provides a link that will launch a User Guide. The intention of the guide is to help a user understand and use the application. A guide can be a generic workflow or a specific step-by-step instructional guide that demonstrates how to interact with the product hosted in the POD application. Since this can be vary from site to site, we provide the ability to alter the content of the User Guide. To edit the User Guide, create your own version of the guide, save it in PDF format using the file name “UserGuide.pdf”. This must be the file name for the user guide link to work properly. Replace the existing pdf on the webserver at <your IIS install folder>\Inetpub\wwwroot\<application folder> with the copy you created, then refresh the application.

This completes any user interface customization that can be done to the POD web application.

Internationalizing POD

The process to internationalize POD is based on guidelines referenced in the "Internationalization with the Dojo Toolkit" found here. It is advisable to get familiar with the process before proceeding. The user interface components in POD including — titles, tooltips, labels and other text are defined in the podconfig.js file. The following steps outline the approach for internationalizing these components by defining resources, then consuming them in the application:

1. Create a folder with the language code for the country inside the nls folder in the web application files. If the defaults were used, then this location will be <your IIS installation folder>\Inetpub\wwwroot\js\nls. For example, if you were going to internationalize POD for Japan, create a folder named "ja" in the location above.
   Note: The nls directory already contains a resource bundle "podi18n.js". This file is the master bundle containing the fall-back strings defined in English.
2. In your language or locale directory created in the previous step, create an identically-named resource bundle as the master bundle and define all the properties and strings applicable in that language. An example for this can be found in the sample "zh" directory available with the POD.
   Note: In the "master" bundle, the properties to be referenced in the application code are declared on the root property, while in the localized Japanese version the properties are at the top level of the object. This is required for internationalization to work correctly.
3. Finally, in the master bundle (the podi18n.js in the root of the /nls folder), add properties that match each locale you have previously defined and set the value of each to true.
   Example

```
define({
    root: {
       // the resources in the master bundle
            },
    "zh": true,
    "ja": true
});
```

4. To consume the resources, replace the existing podconfig.js with podconfig_i18n.js by renaming the file. This version of podconfig file references the properties defined in the resource bundles via dojo/i18n module.
   Note: Only strings that are available with the out-of-the-box version of the application are defined in the sample files. If any customization was performed, those strings must be added to the master bundle and podconfig_i18n.js for internationalization to work correctly.

Appendix

Setting up the Sample Products

There are four example products provided with the POD download. The purpose of these samples is to demonstrate the capability of using the POD web application and ArcGIS for Server to generate maps for the four product types supported through POD. The steps in "Set up ArcGIS for Server for POD", discussed in a previous section, places the samples in the correct location. If defaults were accepted, then the samples will be available on the <ArcGIS Server installation drive>\arcgisserver\MCS_POD\Products. The following sample products are installed:

Type	Product Name	Location
Fixed	Fixed25K	<ArcGIS Server installation drive>\arcgisserver\MCS_POD\Products\Fixed25K
Area	Dynamic Area	<ArcGIS Server installation drive>\arcgisserver\MCS_POD\Products\Dynamic Area
Page Size	Dynamic PageSize	<ArcGIS Server installation drive>\arcgisserver\MCS_POD\Products\Dynamic PageSize
Scale	Dynamic Scale	<ArcGIS Server installation drive>\arcgisserver\MCS_POD\Products\Dynamic Scale

Each sample includes all the required configuration components:
- Map document
- Supporting data
- Python toolbox based geoprocessing tool
- Grid template file
- Other supporting files

In addition to extracting and placing the necessary files in the correct location, follow these steps to use them in the POD web application:

1. **Download additional files for Fixed25K Product**: The folder for fixed 25K contains a document that describes the additional steps to be performed to setup this product correctly. Use the instruction in this document to setup the Fixed25K Product. You will need to download product files for Fixed 25K from GitHub.
2. **Deploy the Map Services**: This step requires ArcGIS Desktop. The map document that will be used to publish the map services that are used in the sample products is located at <ArcGIS Server installation drive>\arcgisserver\MCS_POD\MapServiceData. Open this map

document in ArcMap and publish it to your arcgis server site. Learn more about publishing map service.

3. **Verify the configuration file:** The podconfig.js file provided on GitHub has the necessary properties already defined to enable use of the samples. Make the following updates to the provided config file to work with samples in your environment:
    a. Update the values for Extent Layer service REST URLs to the one published to your server in earlier step.
    ExtentLayers: [
    { value: "25K Index", url: "http://<yourServer>/arcgis/rest/services/mcs_pod/ExtentLayers/MapServer", sublayer: 0, data0: "QUAD_NAME", data1: "SECOORD" },
    { value: "50K Index", url: "http://<yourServer>/arcgis/rest/services/mcs_pod/ExtentLayers/MapServer", sublayer: 1, data0: "NRN" },
    { value: "100K Index", url: "http://<yourServer>/arcgis/rest/services/mcs_pod/ExtentLayers/MapServer", sublayer: 2, data0: "NRN" },
    { value: "Boundaries", url: "http://<yourServer>/arcgis/rest/services/mcs_pod/ExtentLayers/MapServer", sublayer: 3, data0: "STATE_NAME" }],
    b. Uncomment by removing "//" at the beginning of the lines referencing the samples defintions in the ProductDefinitions property. Once done the ProductDefinitions property should look like:
    ProductDefintions: [
    {attrTable: "Fixed25KTable", instanceTable: "Fixed25KInstanceTable"},
    {attrTable: "DynamicAreaTable", instanceTable: "DyanmicAreaInstanceTable"},
    {attrTable: "DynamicScaleTable", instanceTable: "DynamicScaleInstanceTable"},
    {attrTable: "DynamicPageTable", instanceTable: "DynamicPageInstanceTable"}]

This completes the setup for the samples provided with POD. Launch the web site in your browser and use the user guide link to explore these samples.

Discussion on Product Types in POD: Fixed, Area, Scale and Page Size.

All maps display on a flat surface (page or screen), showing some geographic extent at some scale. The following three map characteristics are typically unique for a map but are closely related to each other:

- Geographic extent: The area on the ground that will be displayed in a map.

- Flat viewing surface's extent: The size of the window (or page) that the area on the ground is displayed.
- Map scale: The ratio of the distance on the map to the corresponding distance on the ground.

Given that two out of three characteristics are known, the unknown can be determined by applying a mathematical calculation. Complicating this calculation is the curvature of the earth's surface – the coordinate system of the flat surface as well as the geographic extent's specific latitude and longitude must also be factored into the calculation.

There are four product types supported through POD, depending on the map properties that need to be calculated. Each type is based on how the application determines the unknown map properties given a set of fixed or user-defined values.

The following table lists each product type based on its parameters. Use this table to determine which one fits your needs the best.

Type	Area on Ground	Map Scale	Page Size	Typical use case
Fixed	Fixed	Fixed	Fixed	National Mapping agencies creating maps based on standard specifications for map extents, scale, paper size. Custom Fixed maps are also supported where the area matches a standards but overlaps predefined map extents.
Area	Application will calculate	User-defined	User-defined	Map a point location when the exact areas is unknown but scale, paper size are adjustable
Scale	User defined	Application will calculate	User-defined	Map an area where paper size is restricted so scale adjusts to fit the area on page
Page Size	User-defined	User-defined	Application will calculate	Map an area where paper size is not a restriction so the best fit paper can be calculated for a map.

POD Hardware Architecture Considerations

The hardware requirements for the POD application will vary greatly depending on the amount of traffic and usage on your site. In general, standard ArcGIS for Server deployment patterns should be followed according to your specific needs.

As an example, performance testing has been conducted on a single Amazon EC2 m3.xlarge instance (4 CPU cores, 16 GB RAM) running only ArcGIS Server 10.3, Mapping and Charting Solutions Server 10.3, Microsoft IIS 7.5, and the POD application. Within this environment, approximately 15 concurrent map requests to the 'Fixed 25K' sample product can be fulfilled with only a minor increase in processing time over a single map request. It typically takes about one minute to export

a single PDF of the 'Fixed 25K' product within this environment. However, it is important to note that your custom product's processing times will vary depending on the complexity of your data and MXD, network and database latency, additional load on the server, and a number of other factors. This information is only provided as a high-level frame of reference and is not intended to provide specific recommendations about your particular hardware requirements for a POD deployment.

If you anticipate hosting more complex MXDs, data, and Python scripts on your POD site, you may need to consider a multiple machine ArcGIS for Server deployment to support the increased overhead, as outlined in the deployment scenarios here. In this case, some additional trial and error testing may be necessary to identify a hardware configuration that is suitable for your organization.

The diagrams below depict two possible "production level" options for deploying POD in a distributed environment. The first diagram represents a multiple machine environment and the second represents a single machine configuration.

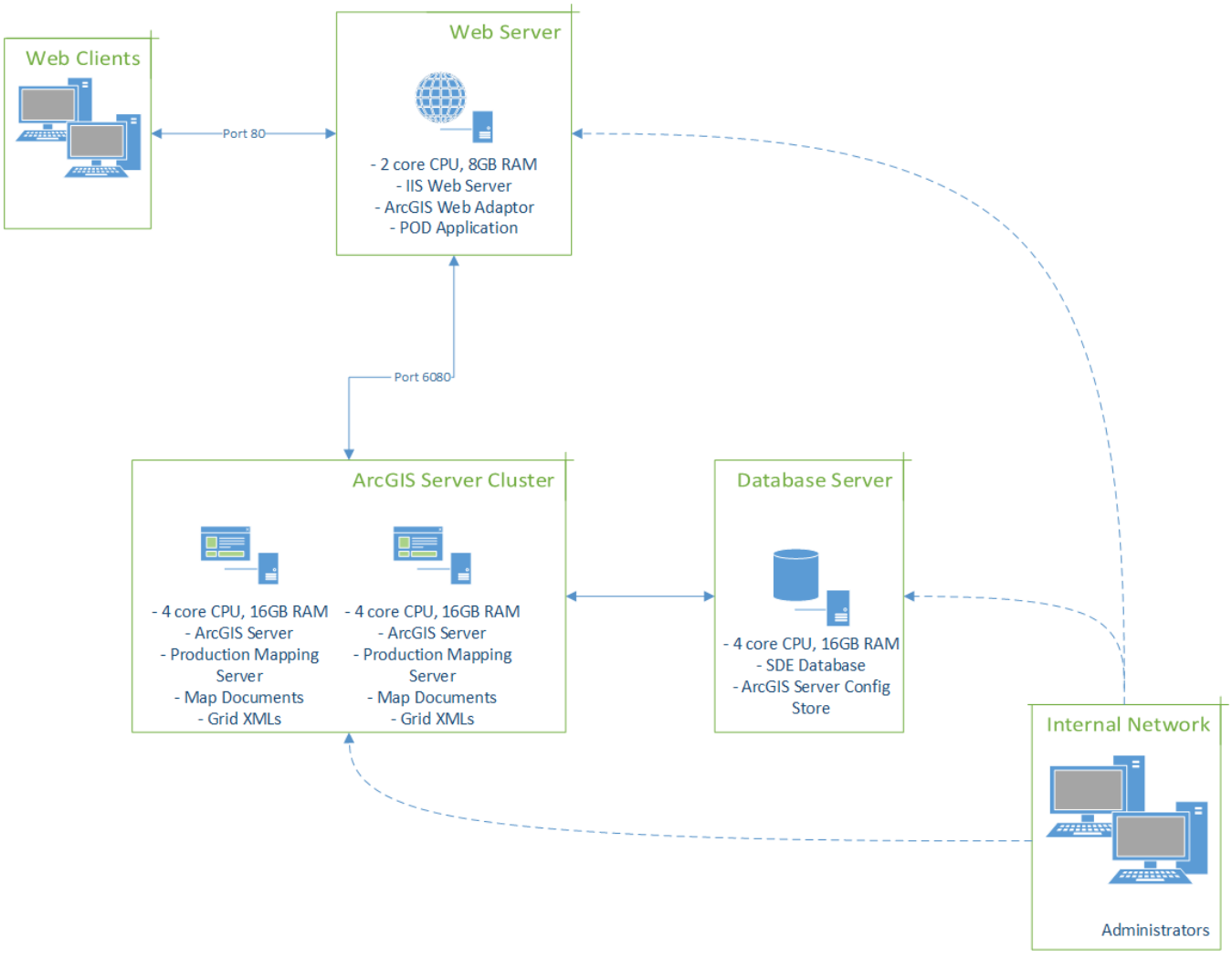


Image 1: Multiple machine configuration

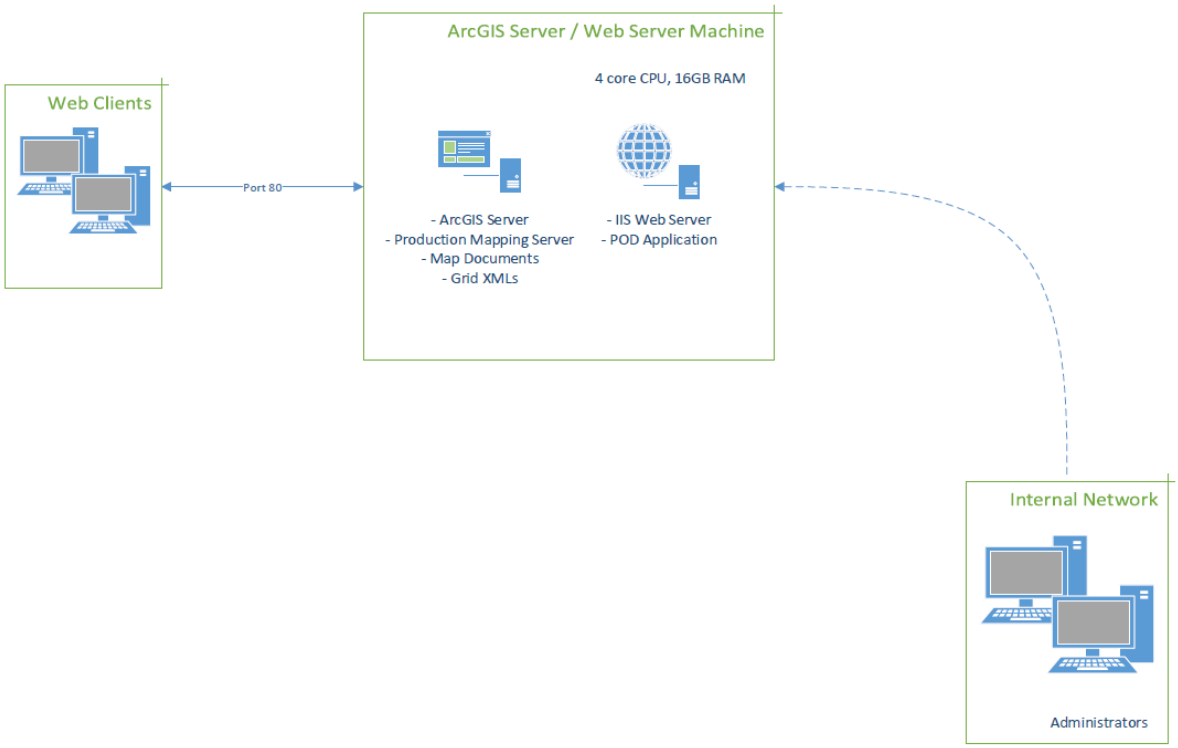


Image 2: Single machine configuration