Service Pack Creation Tool

User Manual

Table of Contents

Service Pack Creation Tool

Introduction

A Service Pack (SP) is a file that comprises of the modifications (i.e. bug fixes and enhancements) over the previous complete released product. It is a file with a **.ppm** extension that can be applied over the base product (or customized application) to incorporate the changes.

The Service Pack Creation tool from WebNMS can be used to create the service pack files. The tool accepts two different versions of product zips to create the service pack.

The Service Pack Creation tool comes along with WebNMS IoT Platform. For more details, contact us at iot-support@webnms.com.

Service Pack Scenario

Consider a scenario where the WebNMS IoT Platform 6.0 customer eg. ABC IoT has customized the product and delivered the initial version of the product to the end customer. Meanwhile ABCIoT receives a service pack update from WebNMS IoT with enhancements and bug fixes over the base WebNMS IoT 6.0 product. To update the product delivered to the end customer with the service pack update, ABCIoT needs to create a service pack over his customized product using the Service Pack Creation tool.

Based on the above scenario, the following steps are involved in creating the service pack. The tool takes two inputs. The main step is creating these two .zip files.

1. Previous complete release. E.g. ABCIoT1.**0** zip
2. Current ppm release with customizations. E.g. ABCIoT1.**1** zip

Service Pack Creation Prerequisite

Procedure to Create ABCIoT1.0.zip

The steps for creating ABCIoT6.0.zip from the base version of WebNMS IoT Platform 6.0 release with customizations is given below.

1. Install WebNMS IoT Platform 6.0 and latest product .ppm (if available for the product). This step is performed on purchasing the WebNMS IoT Platform 6.0 product.
2. Customize the product using Eclipse as per requirement.
3. Ensure that the *<WebNMS IoT Home>/conf/update_conf.xml* has the proper Product Name and Version as given below. If the product has been rebranded, ensure that the Rebranded Product Name and Version is present.

```
<property name="ProductName" value="ABCIoT"/>
<property name="SubProductName" value="BE"/>
<property name="ProductVersion" value="6"/>
<property name="HelpXmlFilePath" value="help/updatehelp.xml"/>
<property name="HelpHtmlFilePath"
value="help/installation_guide/installation_and_setup/installing_service_pack.html"/>
<property name="EnableDeploymentTool" value="true"/>
```

4. Create the .nar using Eclipse. This .nar file will contain the customizations done over Eclipse.
5. Apply the Eclipse .nar file using Deployment Wizard on the fresh base product (in step1).

During WebNMS IoT Service Pack (.ppm) installation, the Product Name and Version in the **Update_conf.xml** file will be verified. If there is a mismatch with the Product Name in the .ppm and the **Update_conf.xml** file, the incompatibility error will be thrown.

6. Create ABCIoT1.0.zip by compressing the <ABCIoT_Home> folder.

Procedure to Create ABCIoT1.1.zip

The steps for creating the ABCIoT1.1.zip after applying the Service Pack (.ppm) from WebNMS IoT is given below:

1. Install WebNMS IoT Platform 6.0.
2. Apply the latest product.ppm.

3. Ensure that the additional customization is present (customization done after creating the ABCIoT1.0.zip).
4. Generate .nar file using Eclipse and deploy it using the Deployment Wizard over the fresh base product (with the .ppm if applicable).
5. Create ABCIoT1.1.zip by compressing the <***ABCIoT_Home***> customized folder.

The following flow diagram illustrates the steps involved in Creating ABCIoT1.1.zip

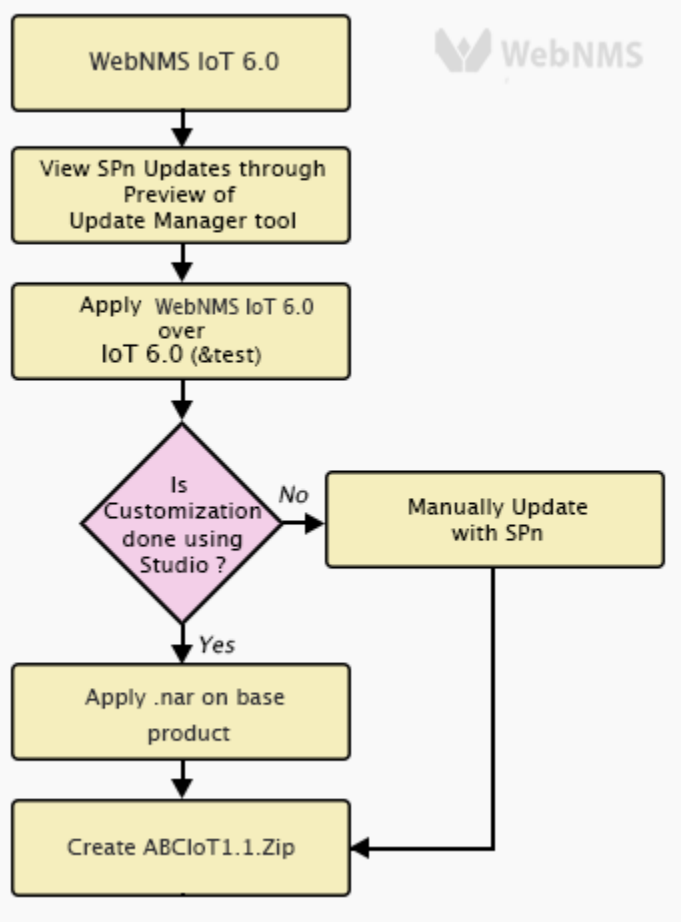

Fig : Flow Diagram

	The naming of the files is very important to maintain the versioning system. It is recommended to have an incremental value in the naming. Eg. base version ABCIoT1.0, enhanced version ABCIoT1.1 zip, next version ABCIoT1.2 zip.

Service Pack Creation

With ABCIoT1.0 and ABCIoT1.1 zip as inputs, the Service Pack Creation tool takes the differences and creates the .ppm, automatically.

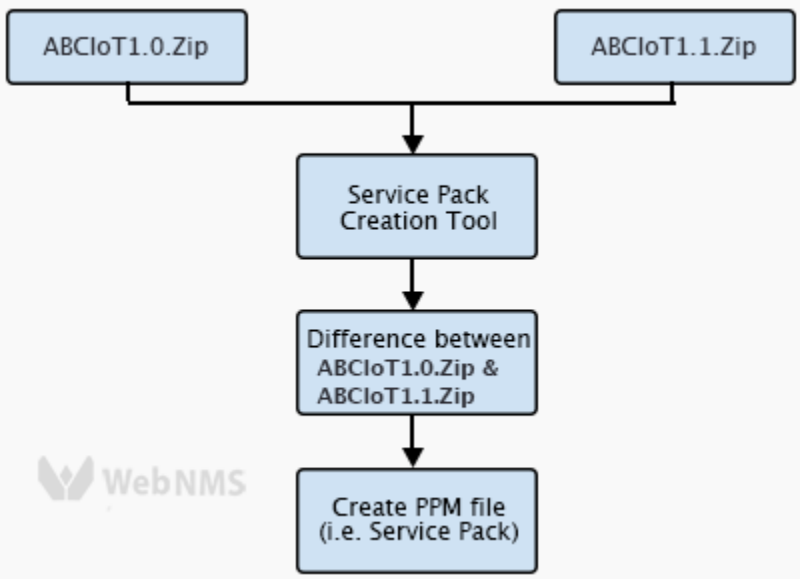

Fig : Service Pack Creation

Initial Configuration

This section details the initial configurations that are required to be performed before using the SP creation tool.

1. Ensure that the mandatory file namely **update_conf.xml** is available in the <*Product Home*>/*conf* directory of both the product zips ABCIoT1.0.zip and ABCIoT1.1.zip. These product zips are given as inputs when running the tool from the command line.

2. The **README** file is mandatory for creating the .ppm file. This file contains the list of enhancements and bug fixes in the .ppm. The absolute path of location of the file must be specified for the **SPReadMe** property in the CreateSP.xml file.

	Do not place files to be packaged for the SP inside the <WebNMS IoT Home>/**Patch** directory. The tool **does not** package the files of the **Patch** directory into the SP or PPM.

Configuring CreateSP.xml and Input.xml file

Apart from the initial configurations, the main configurations need to be performed in the **Input.xml** and **CreateSP.xml** located in the *<SPC tool Home>/conf* folder. It is necessary to understand the tags in these files to proceed with the configuration.

- **Input.xml**
- **CreateSP.xml**

Input.xml File

This configuration file (located in *<SPC tool Home>/conf* directory) is interpreted by the SPC tool for creating the SP (or PPM). It provides input details such as the product name, product version, and SP version which are read and used by the tool to define the name for the created SP. For example, if an SP of the following *product name- **ABCIoT;** product version - **1.0**; and SP version - **1.0*** is to be created, the SP filename will be **ABCIoT_1_0_SP-1_0.ppm.**

The input parameters namely productName and productVersion (product and SP) must be specified correctly in this file, as these are verified and compared with that of the update_conf.xml file of the base product (file inside ABCIoT1.1.zip) during SP installation. If there is a mismatch of the product name or version (either product or SP), then the SP doesn't get applied on the base product.

```
<patchDescription>ABC Test Patch</patchDescription>
<productName>ABCIoT</productName> --name
<productVersion>1.0</productVersion> //Version as in <Webnms IoT Home>/conf/update_conf.xml
<patchVersion>1.0</patchVersion> // Patch version which should incremental compared to the previous patch release.
<patchReadme>ReadMe.html</patchReadme>  // Readme for the end User
```

The explanation of these mandatory input parameters as specified inside the root node i.e., <patchDetails> of the file are as follows:

Tag	Description
patch Description	The description of the SP file is provided in this tag. <patchDescription>**The Test**

	Patch</patchDescription>
productName	The name of the product for which the SP is to be created. For example, if your product name is **ABCIoT** then the entry for this parameter is as shown below. <productName>**ABCIoT**</productName>
productVersion	It is the base product version over which the SP or the PPM is to be applied. The product version specified here must be same as that of the base product over which the created SP or PPM is to be applied. For example, if your base product version is **ABCIoT 1.0** for which you want to create an SP 1.0, then the product version specified here too must be **1.0**. <productVersion>**1.0**</productVersion> **NOTE**: The default product version specified in this Input.xml (located in <SPC tool Home> directory) is **4.0.1**. In this case, you can create SP or PPM only for the base product of version **4.0.1**. Please ensure that you change the default value of the parameter in this file to that of your base product.
SPVersion	It is the version of the created SP or the PPM. This version must be greater than the SP version (if any) of the base product over which it is to be applied. For example, if your base product is of version **4.5.0** and *if you already have an SP with version **1.0*** then the created SP or PPM version must be greater than **1.0** i.e. **1.1**. The entry for this parameter is as given below: <patchVersion>**1.1**</patchVersion>
SPReadme	The name of the SP's Readme file. It is the same file, which is specified for the SPReadMe property in the CreateSP.xml file. For example, if your ReadMe file

	is MyReadMe.html (and whose location is specified in CreateSP.xml file) then the entry for this parameter is as given below. <patchReadme>**MyReadMe.html**</patchReadme>
Updation Context	This tag is used to indicate the context directory viz. BE which is required for SP or PPM creation. By default, the context specified in this file is for a combo setup. <context> <updation context="**BE**" readme="**README.html**" type="**Mandatory**" details="the details about BE context"> </updation> </context>

CreateSP.xml:

This configuration file located in ***<SPC tool Home>/conf*** directory is used by the SPC tool for extracting the modified files from the two zipped products which are provided as inputs to the tool. The various configurable options (i.e., XML definitions) provided in this file enables you to choose or reject the files which you want or don't want to be packaged into the SP. All the XML definitions of this file are as given below:

The main tags in the CreateSP.xml file are the **<createsp>**, **<properties>, <filter>,** and **<classname> tags.**

- **<createsp>**: This is the root mandatory XML tag of the file. All other tags must be specified inside it.
- **<properties>**: This is the main XML tag used for specifying the various properties, which are mandatory for file extraction. The **name** of the property cannot be modified whereas the **value** can be modified.

The definitions of the properties are as follows:

Property	Description
SPDir	This is a mandatory property, which represents the folder (or directory) that stores the

	extracted files of the product zip (i.e., NewZip). You can specify any desired folder name for this property. For example, you can have a folder namely "MyDir" as value for this property, which is as shown here; <property name="SPDir" value="**MyDir**"/> This folder (or directory) is created under the *<SPC Tool Home>* directory. By default, the folder "SPDir" is created inside the *<SPC Tool Home>* directory.
RelativeProductHome	This is a mandatory property which accepts the home path of the zipped products that are provided as input to the tool. The home path of both the input zipped products must be same. For example, if your input zipped products are **ABCIoT1.0.zip** and **ABCIoT1.1.zip** and they both comprise a package structure as ***<ABCIoT_Home>/<all other directories>,*** then their home path is ***<ABCIoT_Home>.*** The value specified for this property is as given below: <property name="RelativeProductHome" value=”**ABCIoT_Home**"/> The relative product home is required as reference for the tool to locate and read the **update_conf.xml** file. The files located in this directory are packaged in the SP or PPM in the same structure as available here.
ResourceDir	This is an optional property which represents the folder (or directory), in which you can place any file (other than those extracted from the product zips) that is to be packaged in the SP or the PPM. For example, assume you want to package a

	file namely **Sample.class** of directory structure "*com/nms/test*". You must then create a resource folder, e.g.MyResourceDir inside <SPC tool Home>, and place the file i.e. **com/nms/test/Sample.class** into this folder. Enter the value for this property as given below: <property name="ResourceDir" value="**MyResourceDir**"/>
SPReadMe	This is a mandatory property which indicates the path of location of the SP readme file. For example, if you have a **MyReadMe.html** in your **C:/winnt/temp** directory, then specify the same path as value for this parameter, as given below: <property name="SPReadMe" value="**C:/winnt/temp/MyReadMe.html**"/> The MyReadMe.html file is copied from the given location to the **SPDir** (located under <*SPC tool Home*>) for packaging it into the SP or the PPM.
InputXML	This is a mandatory property which is used to indicate the path of location of the Input.xml file. By default, the Input.xml file is located in the <*SPC tool Home*>*/conf* directory. But, if the Input.xml file is in a different location, then you must specify its path of location for this property. For example, if the file is located in your **C:/Temp** directory then you need to specify the value for this property as show given below: <property name="InputXML" value="**C:/Temp/Input.xml**"/>

- **<filter>**: This is the filtering XML tag inside which the filtering properties and criteria are specified. The definitions of the filtering properties are given below:
  - **id**: This attribute is used to identify the filter based on whose criteria files are extracted from the product zips. The two types of filters are:

UpdateJarFilter	The advantage of .ujar is to prevent packaging a complete JAR (in a SP or PPM) when it's being updated by just a single file. This is conducive in reducing the packaging size of the created SP or PPM. For example, assume you have added two new files namely Test1.class and Test2.class into a JAR namely **ZAP.jar** (located in <Product Home>/clases directory). To package this ZAP.jar as **.ujar** specify the entry for the property as given below: <filter id="UpdateJarFilter" classname="com.adventnet.tools.update.util.EnhancedFileFilter"> <criterion key="**classes/ZAP.jar**" action="ACCEPT"/> </filter> As per the above entry, the **ZAP.jar** is extracted into the **SPDir** as **ZAP.ujar**. This jar contains only the newly added **.class** files, i.e., Test1.class and Test2.class.The ZAP.jar is packaged with only these two added files, which gets applied on the product during SP or PPM installation.
GenericFilter	This filter is used to segregate files (of the SPDir) which are not to be packaged into the SP or the PPM. This property is handy when you **do not** want to package all the files extracted into the **SPDir**. For example, if you do not want to package the ***tutorials*** and ***help*** files extracted from the product zip into SPDir, then specify them for this filter as given below: <filter id="GenericFilter" classname="com.adventnet.tools.update.util.EnhancedFileFilter"> <criterion key="**tutorials/***" action="REJECT"/> <criterion key="**help/***" action="REJECT"/> </filter> **Note**: The "**/***" after every specified directory for the **criterion** tag means, that all the files of the directory must be rejected for packaging.

- **classname**: This attribute specifies the name of the class, which is invoked for the specific filter type.
  - **<criterion>**: This XML tag is used for setting the criteria for the filters. It comprises two attributes given below:

key	This attribute is to indicate files or folders (along with their complete path of location) which are to be filtered based on the filter type i.e. Generic or UpdateJar filter. For example, if you **do not** want to package an **XYZ.jar**, the entry for this attribute will be as given below: <filter id="UpdateJarFilter" classname="com.adventnet.tools.update.util.EnhancedFileFilter" > <criterion key="**classes/XYZ.jar**" action="REJECT"/> </filter>
action	This attribute is to indicate the action which is to be performed on the files or folders specified for the **key** attribute. Generally, this attribute holds either of the two action types, viz., ACCEPT and REJECT. **ACCEPT**: This action type indicates acceptance of files or folders (specified for the "key" attribute), for packaging into the SP or PPM. **REJECT**: This action type indicates rejection of files or folders (specified for the "key" attribute), from being packaged into the SP or the PPM.

Command Line Options for running the Tool

The options available in the Service Pack Creation tool are listed below:

Command	Description
To create PPM createSP.bat -c OldZip NewZip ConfFile	This command is used to create the SP or the PPM. The SP or the PPM is created based on the inputs of the configuration file namely **CreateSP.xml**. **-c**: It is the switch to create the SP or PPM based on the inputs provided by the configuration file passed for the **confFile** argument. **ConfFile**: It is the configuration file based on whose inputs the the .ppm file is created. Specify the **CreateSP.xml** file for this argument.

	The definition for the **OldZip** and **NewZip** is same as given for the above command. Example Command **creatSP.bat -c IoT6_0.zip IoT6.1.zip conf/CreateSP.xml**
To extract modified and added Files createSP.bat -x OldZip NewZip ConfFile	It is to extract the modified or/and added files and place them in the created SPDir. The files are extracted based on the inputs of the configuration file namely CreateSP.xml. The explanation of the arguments are, **-x**: It is the switch to extract the modified or/and added files from the two product zips and place them in the **SPDir** directory created under the <*SPC tool Home*> directory. The definition for the **OldZip, NewZip,** and **ConfFile** is same as given for the above command. Example Command **createSP.bat -x IoT6_0.zip IoT6.1.zip conf/CreateSP.xml**
To retrieve the list of modified and added files createSP.bat -l OldZip NewZip	It is to retrieve the list of modified or/and added files contained in both the product zips (given as inputs to the tool). The explanation of the arguments are, **-l**: It is the switch to list the modified or/and added files **OldZip**: It is the base product zip (mostly the SP is appled over it e.g. **JSR_1_0.zip**). **NewZip**: It is the latest product zip containing the customizations performed over the base product e.g.**JSR_1_0.zip** Example Command: **createSP.bat - l IoT6_0.zip IoT6.1.zip** **Note**: You can maintain copy of the modified files in

	any file e.g. **a.txt**. The a.txt file is created inside the *<SPC tool Home>/bin* directory with all the modified files. Example command: **createSP.bat - l IoT6_0.zip IoT6.1.zip >a.txt**