The article provides instructions for enabling the IBM® WebSphere® Application Server to use IBM Software Development Kit, Java™ Technology Edition Version 7.0.

View more content in this series

Introduction

IBM Software Development Kit, Java Technology Edition V7.0 (SDK 7) is supported as an optional, pluggable software development kit for IBM WebSphere Application Server V8.5, while IBM Software Development Kit, Java Technology Edition V6.0 (SDK 6) remains the default SDK.

Earlier versions of WebSphere Application Server only supported one version of the SDK. For example, WebSphere Application Server Versions 7.0 and 8.0 both support SDK 6. After moving up to WebSphere Application Server V8.5 from either of those versions, you can continue running your legacy applications on SDK 6, or you can optionally run them on SDK 7.

A pluggable SDK is a version of a Java software developer kit that you can optionally use with WebSphere Application Server. For WebSphere Application Server V8.5, the pluggable SDKs are IBM WebSphere Software Development Kit, Java Technology Edition V7.0 (on System z® and distributed platforms) and IBM Technology for Java 7.0 (on System i®).

WebSphere Application Server V8.5 contains information about SDKs that it supports (SDK name, version, bit architecture, location, and so on). As the default SDK, SDK 6 is installed with WebSphere Application Server V8.5 on z/OS® and the distributed platforms, and is separately

Try WebSphere Application Server V8.5 now

Try IBM WebSphere Application Server V8.5 trial version free for 60 days, or get IBM WebSphere Application Server for Developers V8.5 now at no cost, and with no expiration date!
installed on IBM i systems. When an optional SDK, such as SDK 7, is installed, it is detected by the product and becomes available for use; likewise, if an optional SDK is uninstalled, it is no longer detected and is no longer available. Enabling WebSphere Application Server to use an available SDK is accomplished using the commands and interfaces described below.

Enabling WebSphere Application Server to use SDK 7

First, you need to install SDK 7. Once that is complete, you will then be able to:

- List the available SDKs
- Enable commands to use SDK 7 as the command default SDK
- Enable new profiles to use SDK 7
- Enable standalone application server profiles and management profiles to use SDK 7
- Enable managed nodes/servers/clusters to use SDK 7
- Enable a new application server, cluster member template, and new cluster member to use SDK 7
- Enable all servers and commands to use SDK 7
- Enable the application client to use SDK 7

The sections that follow include details and instructions for performing these tasks.

Install SDK 7

IBM WebSphere SDK Java Technology Edition Version 7.0 is installed using the IBM Installation Manager on z/OS and on the distributed operating systems. See Installing and un installing SDK Java Technology Edition Version 7.0 in the WebSphere Application Server V8.5 Information Center for complete instructions.

For System i, SDK 7 is available as 5761JV1 option 14 (Java SE 7 32 bit) and option 15 (Java SE 7 64 bit) for IBM System i 7.1; however, Java SE 7 is not available for IBM System i 6.1. For more information, see the WebSphere Application Server V8.5 Information Center for System i prerequisites.

After installing SDK 7, restart all of these server types to trigger updates to the node metadata properties for SDK 7:

- Node agent server
- Standalone application server
- Management server (deployment manager server, admin agent server, or a job manager server)

Similarly, after uninstalling an SDK, you must restart these server types to trigger removal of the information associated with the uninstalled SDK from the node metadata properties.

List the available SDKs

Listing the available SDKs enables you to determine the name of an available SDK 7 installation so you can tell WebSphere Application Server to use that installation.
Use the `managesdk -listAvailable` command with the `-verbose` option to list the name of each available SDK and its associated property values (Listing 1).

Listing 1

```
C:\wasInstalls\v85\Base\bin>managesdk.bat -listAvailable -verbose
CWSDK1003I: Available SDKs :
CWSDK1005I: SDK name: 1.6_32
  - com.ibm.websphere.sdk.version.1.6_32=1.6
  - com.ibm.websphere.sdk.bits.1.6_32=32
  - com.ibm.websphere.sdk.location.1.6_32=${WAS_INSTALL_ROOT}/java
  - com.ibm.websphere.sdk.platform.1.6_32=windows
  - com.ibm.websphere.sdk.architecture.1.6_32=x86_32
  - com.ibm.websphere.sdk.nativeLibPath.1.6_32=${WAS_INSTALL_ROOT}/lib/native/win/x86_32/
CWSDK1005I: SDK name: 1.7_32
  - com.ibm.websphere.sdk.version.1.7_32=1.7
  - com.ibm.websphere.sdk.bits.1.7_32=32
  - com.ibm.websphere.sdk.location.1.7_32=${WAS_INSTALL_ROOT}/java_1.7_32
  - com.ibm.websphere.sdk.platform.1.7_32=windows
  - com.ibm.websphere.sdk.architecture.1.7_32=x86_32
  - com.ibm.websphere.sdk.nativeLibPath.1.7_32=${WAS_INSTALL_ROOT}/lib/native/win/x86_32/
CWSDK1001I: Successfully performed the requested managesdk task.
```

In the above example, the SDK name of the SDK 7 installation is 1.7_32. We know this because the value of the `com.ibm.websphere.sdk.version.1.7_32` property is 1.7, but not because the SDK name happens to include the version (1.7). Even though all currently supported SDK names follow the pattern of `<version>_<bitness>`, there is no formal convention for supported SDK names. Therefore, it is good practice to look it up using the `managesdk -listAvailable` command with the `-verbose` option. If you do not use the `-verbose` option, then only the SDK names are listed.

Enable commands to use SDK 7 as the command default SDK

After setting the command default SDK to SDK 7, commands will use SDK 7 when issued from the bin directory under the WebSphere Application Server installation root directory (`<was_install_root>/bin`), unless a valid profile name is specified on the command line (such as, `-profileName myprofile`) or a default profile exists, in which case the command default SDK is overridden by the SDK enabled for the specified or default profile, respectively. However, commands invoked from the bin directory under a WebSphere Application Server profile root directory (`<profile_root>/bin`) always use the SDK enabled for that profile.

Before beginning, list the available SDKs to determine the SDK name of an available SDK 7 installation, then use the `managesdk -setCommandDefault` command with the `-sdkName` option to enable commands to use SDK 7 (Listing 2).

Listing 2

```
C:\wasInstalls\v85\Base\bin>managesdk.bat -setCommandDefault
-sdkName 1.7_32
CWSDK1021I: The command default SDK name is now set to 1.7_32.
CWSDK1001I: Successfully performed the requested managesdk task.
```
Enable new profiles to use SDK 7

After enabling new profiles to use SDK 7, all profiles you subsequently create are automatically initially enabled to use SDK 7.

Before beginning, list the available SDKs to determine the SDK name of an available SDK 7 installation, then use the managesdk -setNewProfileDefault command with the -sdkName option to enable new profiles to use SDK 7. The example in Listing 3 assumes that the SDK name of an available SDK 7 installation is 1.7_32.

Listing 3

C:\wasInstalls\v85\Base\bin>managesdk.bat -setNewProfileDefault
-sdkName 1.7_32
CWSDK1022I: New profile creation will now use SDK name 1.7_32.
CWSDK1001I: Successfully performed the requested managesdk task.
C:\wasInstalls\v85\Base\bin>

Enable standalone application server profiles and management profiles to use SDK 7

These profile types fall into this category:

- Standalone application server profile
- Custom profile
- Management profile, including deployment manager profile, administrative agent profile, and job manager profile

You can use one of the following methods to enable SDK 7 for such profiles:

a. Using the managesdk command

Before you begin:

1. Stop the affected server.
2. List the available SDKs to determine the name of an available SDK 7 installation.

Invoke the managesdk -enableProfile command with the -enableServers option to enable a standalone application server profile or management profile to use SDK 7. For example, assuming the SDK name of an available SDK 7 installation is 1.7_32 and profile “server1” defines a standalone or management server, invoke managesdk as shown in Listing 4 to ensure profile server1 and the server it defines is enabled to use SDK 7.

Listing 4

>cd <WAS_INSTALL_ROOT>\bin
$>
>managesdk.bat -enableProfile
-profileName server1
-sdkName 1.7_32
-enableServers
CWSDK1017I: Profile server1 now enabled to use SDK 1.7_32.
CWSDK1001I: Successfully performed the requested managesdk task.
Additionally, if administrative security is enabled in the cell, you can provide your userid and password on the command by using the `-user` and `-password` options; otherwise, you can configure security for scripting. See the WebSphere Application Server V8.5 Information Center for details.

b. **Using AdminTask SDK commands**

There are two AdminTask SDK commands that can be used to enable a standalone server profile to use SDK 7, one at a node level and another at the server level.

**To enable a standalone server profile to use SDK 7 at a node level** using AdminTask.setNodeDefaultSDK() for a given node:

1. Stop the server.
2. From the bin directory of the standalone server profile, start the wsadmin tool in a local mode (Listing 5).

**Listing 5**

```
cd <profile_directory>/bin
c:\<profile_directory>\bin> wsadmin.bat -conntype none -lang jython
```

3. Find available SDK names to use for a given node (the AdminTask command in Listing 6 only shows a newly installed SDK as an available SDK for a node after the SDK information is collected as part of its node metadata properties).

**Listing 6**

```
wsadmin>print AdminTask.getAvailableSDKsOnNode('[-nodeName myNodeName]')
1.6_32
1.7_32
```

4. Under wsadmin prompt, run the `AdminTask.setNodeDefaultSDK()` command (Listing 7).

**Listing 7**

```
wsadmin>AdminTask.setNodeDefaultSDK('[-nodeName myNode -sdkName 1.7_32 --clearServerSDKs true]''
wsadmin>AdminConfig.save()
```

5. Optionally, validate the node default SDK is set to 1.7_32 (Listing 8).

**Listing 8**

```
wsadmin>AdminTask.getNodeDefaultSDK('[-nodeName myNode]
'[['javahome ${JAVA_LOCATION_1.7_32}] [sdkname 1.7_32]]
wsadmin>quit
```

6. After the node is successfully set to use SDK 7, restart the server to run SDK 7.

**Be aware:**

- The `-clearServerSDKs` parameter of the `AdminTask.setNodeDefaultSDK()` command clears the server level SDK setting and enables the server to use the node level default SDK.
- If the node specified is a deployment manager node, only the deployment manager server is enabled to use the node level default SDK.
To enable a standalone server profile to use SDK 7 at a server level using AdminTask.setServerSDK():

1. Stop the server.
2. From the bin directory of the standalone server profile, start the wsadmin tool in a local mode (Listing 5).

```
Listing 9

cd <profile_directory>/bin
c:<profile_directory>\bin> wsadmin.bat -conntype none -lang jython
```

3. Find available SDK names to use for its parent node (the AdminTask command in Listing 10 only shows a newly installed SDK as an available SDK for a node after the SDK information is collected as part of its node metadata properties).

```
Listing 10

wsadmin>print AdminTask.getAvailableSDKsOnNode('[-nodeName myNodeName]')
1.6_32
1.7_32
```


```
Listing 11

wsadmin>AdminTask.setServerSDK('[-nodeName myNode -serverName myServer
   -sdkName 1.7_32]')
''
wsadmin>AdminConfig.save()
```

5. Optionally, validate the server SDK is set to 1.7_32 (Listing 12).

```
Listing 12

wsadmin>AdminTask.getServerSDK('[-nodeName myNode -serverName myServer]')
'[[javahome ${JAVA_LOCATION_1.7_32}] [sdkname 1.7_32]]'
wsadmin>quit
```

6. After the node is successfully set to use SDK 7, restart the server to run in SDK 7.

c. Using administrative console:

SDK 7 should already be installed on the node with the server that you wish to update. The nodeAgent or single server must be restarted so that the node-metadata is available.

To enable SDK 7 on a single server, whether standalone or in a Network Deployment (ND) environment, navigate to the server detail panel (Servers > Server Types > Application Server > ServerName), and select the Java SDKs link in the Server Infrastructure section (Figure 1).
If the SDK 7 was installed correctly and the node agent or server restarted, you should see Figure 2 when **Java SDKs** is selected.

**Figure 2. Description?**

Select the SDK 7 installation and click the **Make Default** button to make SDK 7 the default SDK. When the console displays the Save or Review links, choose **Save**. If running in an ND environment and the console preference to automatically synchronize is not set, the node
must be manually synchronized. After synchronization, the server must be restarted for the change to take effect.

The AdminTask commands that the console issues to modify and display the list of available SDKs are shown in Figure 3.

**Figure 3. Description?**

<table>
<thead>
<tr>
<th>Administrative Scripting Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminTask.setServerSDK([&quot;nodeName IBM-E785F072XCLNode065&quot;, &quot;servername server1&quot;, &quot;sdkName 1.7_64&quot;]</td>
<td>Enable managed nodes/servers/clusters to use SDK 7</td>
</tr>
<tr>
<td>AdminTask.getAvailableSDKsOnNode([&quot;nodeName IBM-E785F072XCLNode065&quot;]</td>
<td>Prior to enabling a managed node to use SDK 7 after SDK 7 is installed, all node agent servers within the WebSphere Application Server installation should be stopped and restarted while the deployment manager server is running. This will help to trigger the newly installed SDK 7 information to be added in as part of the node metadata properties, which makes SDK 7 available for a node to use.</td>
</tr>
<tr>
<td>AdminTask.getSDKPropertiesOnNode([&quot;nodeName IBM-E785F072XCLNode065&quot;, &quot;sdkName 1.6_64&quot;, &quot;sdkAttributes [version location bit]&quot;</td>
<td>To configure SDK 7 for a federated profile or node, use the managesdk command or AdminTask.setNodeDefaultSDK() command.</td>
</tr>
<tr>
<td>AdminTask.getSDKPropertiesOnNode([&quot;nodeName IBM-E785F072XCLNode065&quot;, &quot;sdkName 1.7_64&quot;, &quot;sdkAttributes [version location bit]&quot;</td>
<td>a. Using managesdk command</td>
</tr>
<tr>
<td>AdminTask.getSDKPropertiesOnNode([&quot;nodeName IBM-E785F072XCLNode065&quot;, &quot;sdkName 1.7_64&quot;, &quot;sdkAttributes [version location bit]&quot;</td>
<td>Before you begin:</td>
</tr>
<tr>
<td>AdminTask.getSDKPropertiesOnNode([&quot;nodeName IBM-E785F072XCLNode065&quot;, &quot;sdkName 1.7_64&quot;, &quot;sdkAttributes [version location bit]&quot;</td>
<td>1. Start the node agent.</td>
</tr>
<tr>
<td>AdminTask.getSDKPropertiesOnNode([&quot;nodeName IBM-E785F072XCLNode065&quot;, &quot;sdkName 1.7_64&quot;, &quot;sdkAttributes [version location bit]&quot;</td>
<td>2. List the available SDKs to determine the name of an available SDK 7 installation.</td>
</tr>
<tr>
<td>AdminTask.getSDKPropertiesOnNode([&quot;nodeName IBM-E785F072XCLNode065&quot;, &quot;sdkName 1.7_64&quot;, &quot;sdkAttributes [version location bit]&quot;</td>
<td>3. Ensure the deployment manager is running. The SDK configuration change to the federated profile is made to the deployment manager's master configuration repository, and not directly to the federated profile's configuration.</td>
</tr>
</tbody>
</table>

Invoke the `managesdk -enableProfile` command with the `-enableServers` option to enable a federated profile to use SDK 7. For example, assuming the SDK name of an available SDK 7 installation is 1.7_32 and profile fedProfile1 is the federated profile to be enabled to use SDK 7, invoke the `managesdk -enableProfile` command as shown in Listing 13 to ensure the fedProfile1 profile and the servers on the node it defines are enabled to use SDK 7:
Listing 13

```bash
> managesdk.bat -enableProfile
  -profileName fedProfile1
  -sdkName 1.7_32
  -enableServers
CWSDK1024I: The node default SDK setting for federated profile fedProfile1 has been saved in the master configuration repository.
CWSDK1025I: A synchronization operation is required before configuration changes to federated profile fedProfile1 can be used.
CWSDK1017I: Profile fedProfile1 now enabled to use SDK 1.7_32.
CWSDK1001I: Successfully performed the requested managesdk task.
```

Additionally, if administrative security is enabled in the cell, you can provide your userid and password on the command by using the `–user` and `–password` options; otherwise; you can configure security for scripting. See the [WebSphere Application Server V8.5 Information Center](https://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/ctscfg.html) for details.

Before an application server in a federated node can be used with this new SDK setting, you must perform a synchronization operation.

b. **Using AdminTask.setNodeDefaultSDK() command**

1. Start the deployment manager server of a managed node, if it is not already running.
2. From the bin directory of the deployment manager server profile, start the wsadmin tool in a connected mode (Listing 14).

**Listing 14**

```bash
cd <profile_directory>/bin
\c: //<profile_directory>//bin> wsadmin.bat -conntype SOAP –port dmgr_soap_port
  -lang jython
```

3. Check for available SDKs to use by running the commands shown in Listing 15 (to find available SDK names for a node) and Listing 16 (to find available SDK install locations for a node).

**Listing 15**

```bash
wsadmin>print AdminTask.getAvailableSDKsOnNode('[-nodeName myNodeName]')
1.6_32
1.7_32
```

**Listing 16**

```bash
wsadmin>AdminTask.getSDKPropertiesOnNode('[-nodeName myNodeName -sdkName 1.7_32
  -sdkAttributes [location ]]')
'[com.ibm.websphere.sdk.location.1.7_32 ${WAS_INSTALL_ROOT}/java_1.7_32]'
```

4. Under a wsadmin prompt, run the `AdminTask.setNodeDefaultSDK()` command (Listing 17 or 18).

**Listing 17**

```bash
wsadmin>AdminTask.setNodeDefaultSDK('[-nodeName myNode -sdkName 1.7_32
  -clearServerSDKs true]')
''
wsadmin>AdminConfig.save()
```
5. Optionally, validate the node default SDK is set to 1.7_32 (Listing 19 or 20).

**Listing 19**

```bash
wsadmin>AdminTask.getNodeDefaultSDK('[-nodeName myNode]')
'[[javahome ${JAVA_LOCATION_1.7_32}] [sdkname 1.7_32]]'
```

**Listing 20**

```bash
wsadmin>AdminTask.getNodeDefaultSDK('[-nodeName myNode]')
'[[javahome ${WAS_INSTALL_ROOT}/java_1.7_32] [sdkname 1.7_32]]'
```

6. By default, the SDK change to the node will be propagated periodically to the affected node as long as the node can communicate with the deployment manager. You can also run the script in Listing 21 right after step 4 to enforce that the SDK change be manually propagated to the affected nodes.

**Listing 21**

```bash
wsadmin>AdminNodeManagement.syncNode("myNode")
wsadmin>quit
```

7. After the node is successfully set to use SDK 7, all servers must be restarted to run in SDK 7.

Be aware that the -clearServerSDKs parameter of the AdminTask.setNodeDefaultSDK() command clears the server level SDK setting and enables the server to use the node level default SDK.

**To configure SDK 7 for a server in a managed node**, use the AdminTask.setServerSDK() command or an admin console:

a. **Using AdminTask.setServerSDK() command**

1. Start the deployment manager server of a managed node, if it is not already running.
2. From the bin directory of the deployment manager server profile, start the wsadmin tool in a connected mode (Listing 22).

**Listing 22**

```bash
cd <profile_directory>/bin
c:\<profile_directory>\bin> wsadmin.bat -conntype SOAP -port dmgr_soap_port -lang jython
```

3. Check for available SDKs to use by running the commands in Listing 23 (to find available SDK names for a node) and Listing 24 (to find available SDK install locations for a node).
Listing 23

```
wsadmin> print AdminTask.getAvailableSDKsOnNode('[-nodeName myNodeName]')
1.6_32
1.7_32
```

Listing 24

```
wsadmin> AdminTask.getSDKPropertiesOnNode('[-nodeName myNodeName -sdkName 1.7_32 -sdkAttributes [location]]')
'[com.ibm.websphere.sdk.location.1.7_32 ${WAS_INSTALL_ROOT}/java_1.7_32]'
```

4. Under a wsadmin prompt, run the `AdminTask.setServerSDK()` command (Listing 25 or 26).

Listing 25

```
wsadmin> AdminTask.setServerSDK('[-nodeName myNode -serverName myServer -sdkName 1.7_32]')
```

Listing 26

```
wsadmin> AdminTask.setServerSDK('[-nodeName myNode -serverName myServer -javahome ${WAS_INSTALL_ROOT}/java_1.7_32]')
'null'
```

5. Optionally, validate that the server SDK is set to 1.7_32 (Listing 27 or 28).

Listing 27

```
wsadmin> AdminTask.getServerSDK('[-nodeName myNode -serverName myServer]')
'[javahome ${JAVA_LOCATION_1.7_32}] [sdkname 1.7_32]'
```

Listing 28

```
wsadmin> AdminTask.getServerSDK('[-nodeName myNode -serverName myServer]')
'[javahome ${WAS_INSTALL_ROOT}/java_1.7_32] [sdkname 1.7_32]'
```

6. By default, the SDK change to the node will be propagated periodically to the affected node as long as the node can communicate with the deployment manager. You can also run the script in Listing 29 right after step 4 to enforce that the SDK change be manually propagated to the affected nodes.

Listing 29

```
wsadmin> AdminNodeManagement.syncNode("myNode")
wsadmin> quit
```

7. After the server is successfully set to use SDK 7, restart the server to run in SDK 7.

b. **Using the admin console**

SDK 7 should already be installed on the node that you wish to update. The nodeAgent must be restarted so that the node-metadata is available.
1. **To enable SDK 7 on a managed node**, navigate to the node collection panel (System Administration > Nodes > NodeName), and select the Java SDKs link in the Additional Properties section (Figure 4).

**Figure 4. Description?**

![Node configuration panel](image)

1. To enable SDK 7 on a managed node, navigate to the node collection panel (System Administration > Nodes > NodeName), and select the Java SDKs link in the Additional Properties section (Figure 4).

2. If the SDK 7 was installed correctly and the node agent is restarted, you should see Figure 5 when Java SDKs is selected.

**Figure 5. Description?**

![Java SDKs configuration](image)

Select the SDK 7 installation and click the **Make Default** button to make SDK 7 the default SDK. After the default is changed, a message indicating success or failure is displayed (Figure 6).
3. When the console displays the Save or Review links, choose **Save**. If the console preference to automatically synchronize is not set, the node must be synchronized manually. After the synchronization, the node agent must be restarted for the change to take effect.

The AdminTask commands that the console issues to modify and display the list of available SDKs are shown in Figure 7.

**Figure 7. Description?**

To configure **SDK 7 for all cluster members in a cluster**, use `AdminTask.setServerSDK()` command:
1. Start the deployment manager server of a managed node, if it is not already running.
2. From the bin directory of the deployment manager server profile, start the wsadmin tool in a connected mode (Listing 30).

Listing 30

cd <profile_directory>/bin

c:\<profile_directory>\bin> wsadmin.bat -conntype SOAP -port dmgr_soap_port -lang jython

Check available SDKs for its patent node. To find available SDK names for a node use the command shown in Listing 31. To find available SDK install locations for a node, use the command shown in Listing 32.

Listing 31

wsadmin>print AdminTask.getAvailableSDKsOnNode('[-nodeName myNodeName]')
1.6_32
1.7_32

Listing 32

wsadmin>AdminTask.getSDKPropertiesOnNode('[-nodeName myNodeName -sdkName 1.7_32 -sdkAttributes [location ]]
'[com.ibm.websphere.sdk.location.1.7_32 ${WAS_INSTALL_ROOT}/java_1.7_32]

3. Under a wsadmin prompt, run the AdminTask.setServerSDK() command (Listing 33 or 34).

Listing 33

wsadmin>AdminTask.setServerSDK('[-clusterName myCluster -sdkName 1.7_32]')

wsadmin>AdminConfig.save()

Listing 34

wsadmin>AdminTask.setServerSDK('[-clusterName myCluster -javahome ${WAS_INSTALL_ROOT}/java_1.7_32]')

wsadmin>AdminConfig.save()

4. Optionally, validate the cluster member SDK is set to 1.7_32 (Listing 35 or 36).

Listing 35

wsadmin>AdminTask.getServerSDK('[-nodeName myNode –servername myClusterMember]')
'[[javahome ${JAVA_LOCATION_1.7_32}] [sdkname 1.7_32]]'

Listing 36

wsadmin>AdminTask.getServerSDK('[-nodeName myNode –servername myClusterMember]')
'[[javahome ${WAS_INSTALL_ROOT}/java_1.7_32] [sdkname 1.7_32]]'

5. By default, the SDK change to the node will be propagated to the affected node periodically as long as the node can communicate with the deployment manager. You can also run the
script in Listing 37 right after step 4 to enforce that the SDK change is propagated manually to the affected nodes.

Listing 37

```
wsadmin>AdminNodeManagement.syncNode("myNode")
wsadmin>quit
```

After the cluster is successfully set to use SDK 7, all cluster member servers must be restarted to use SDK 7.

**Enable a new application server, cluster member template, and new cluster member to use SDK 7**

- **New application server**
  When an application server is created, there is no SDK setting at the server level; it inherits its node SDK setting by default. If its node SDK is set to SDK 7, the new application server also uses SDK 7; otherwise, you can use the `AdminTask.setNodeDefaultSDK()` or `AdminTask.setServerSDK()` command to set the new application server to use SDK 7 at a node level or a server level.

- **Cluster member template**
  You can use either of these two procedures to enable a cluster member template to use SDK 7:
  a. Using the `AdminTask.setServerSDK()` command with `-clusterName` parameter not only enables SDK 7 for all cluster members, but also enables its cluster member template to use SDK 7 so that all cluster members created subsequently will also be enabled to use SDK 7.
  b. You can create an application server that has SDK 7 enabled at the server level, and then create the first cluster member either by using that application server as a cluster member template, or by converting that application server to the first cluster member. After the first cluster member is created, the cluster member template has SDK 7 enabled at the server level.

- **Cluster and new cluster members**
  Within a cluster, the SDKs used by all cluster members must comply by being of the same version and same number of bits.

  When a cluster member is created, if its server template has no SDK setting at a server level, it will use the node level of SDK setting by default. If the node SDK setting is not what you want, you can use one of the methods described earlier to enable the node to use the SDK that you want.

  When a cluster member is created, if it uses a server template that has the SDK setting at a server level, the cluster member uses the server SDK setting, which takes a precedent over the node SDK setting. For example, if the server template is enabled to use SDK 7, the cluster member created using that server template is also enabled to use SDK 7, regardless of the SDK setting at its node level.

  To see the SDK setting for a newly created cluster member, run the `AdminTask` command in Listing 38.
Listing 38

```java
wsadmin>AdminTask.getServerSDK('[-nodeName myNode -serverName myClusterMember]')
  '[javahome ${WAS_INSTALL_ROOT}/java_1.7_32 [sdkname 1.7_32]]'
wsadmin>
```

If the cluster member uses `${WAS_INSTALL_ROOT}/java_1.7_32` value for JAVA_HOME at a server SDK setting, a user needs to make sure that an SDK 7 in 32 bits is installed and is available for the parent node of that cluster member at the specified location. For IBM System i 7.1, you need to make sure that the SDK location set for JAVA_HOME has the corresponding SDK installed.

If the cluster member uses `${JAVA_LOCATION_1.7_32}` value for JAVA_HOME at a server setting, it is your responsibility to make sure that the parent node of the cluster member has already been enabled to use SDK 7, which sets the value of `${JAVA_LOCATION_1.7_32}`.

### Enable all servers and commands to use SDK 7

To enable a new WebSphere Application Server V8.5 installation to use SDK 7 for all servers and commands prior to creating any profiles, simply follow the earlier instructions to:

1. Enable commands to use SDK 7 as the command default SDK
2. Enable new profiles to use SDK 7

If you are changing the SDK configuration for a V8.5 installation that has existing profiles, then in addition to the two tasks listed above, you will also need to enable all existing profiles to use SDK 7.

Before you begin:

1. Stop all application servers.
2. Start all node agents.
3. List the available SDKs to determine the name of an available SDK 7 installation.
4. Ensure the deployment managers for all federated nodes are running. The SDK configuration change for federated nodes are made to the deployment manager's master configuration repository, and not directly to the federated profile's configuration.

To enable all the existing profiles to use SDK 7, invoke the `managesdk -enableProfileAll` command with the `-sdkName` option. For example, if the `sdkName` of an SDK 7 installation is `1.7_32`, then invoke `managesdk` as shown in Listing 39.

#### Listing 39

```bash
>cd <WAS_INSTALL_ROOT>/bin
> managesdk.bat -enableProfileAll
  -sdkName 1.7_32
  -enableServers
CWSDK1017I: Profile default now enabled to use SDK 1.7_32.
CWSDK1017I: Profile server1 now enabled to use SDK 1.7_32.
CWSDK1017I: Profile appServer2 now enabled to use SDK 1.7_32.
CWSDK1001I: Successfully performed the requested managesdk task.
```
Additionally, if administrative security is enabled, you can provide your userid and password on the command by using the -user and -password options. However, if you cannot authenticate to all cells with the same userid and password, you must configure security for scripting to use -enableProfileAll. See the WebSphere Application Server V8.5 Information Center for information on how to configure security for scripting.

Before an application server in a federated node can be used with this new SDK setting, a synchronization operation must be performed.

To avoid trouble, do not use the -enableProfileAll option unless automatic prompting is disabled for SOAP, IPC, and RMI connections to the deployment managers of cells that have any federated node or deployment manager with security enabled. Automatic prompting causes the managesdk command to fail or stop processing.

**Enable the application client to use SDK 7**

Enabling application client installations on distributed platforms to use SDK 7 can be achieved by simply enabling commands to use SDK 7 as the command default SDK.

On the System i, application client installations can contain profiles. Therefore, in addition to enabling commands to use SDK 7 as the command default SDK, you must enable new profiles to use SDK 7. Also, if profiles exist, use the managesdk -enableProfileAll command to enable all existing profiles to use SDK 7. For example, if the SDK name of the SDK 7 installation is 1.7_32 and the application client contains only a single profile named client1, then managesdk can be invoked as shown in Listing 40.

### Listing 40

```
> managesdk -enableProfileAll -sdkName 1.7_32
CWSDK1017I: Profile client1 now enabled to use SDK 1.7_32.
CWSDK1001I: Successfully performed the requested managesdk task.
```

**Commands and interfaces**

**The managesdk command**

WebSphere Application Server V8.5 provides a command line interface in the form of a script called `managesdk`, which enables administrators to manage the various SDK configuration settings for their installations. It is profile-centric in its function, and can be called from either the bin directory of the product installation or the bin directory of any profile.

The managesdk script calls setter and getter tasks that change or retrieve SDK configuration settings related to the installation itself, a specified profile, or to all profiles in an installation. You can invoke managesdk tasks to list the current SDK settings for a single profile (or all profiles) to enable a single profile (or all profiles) in an installation to use an available SDK, or to list all available SDKs. Additionally, you can invoke managesdk tasks to get and set the SDK that is used by non-profile dependent commands.
Some of the tasks supported by the managesdk command were already mentioned in the Introduction, but there are also three optional parameters to the managesdk command worth noting:

- **-debug**: Can be added to any managesdk task request. It provides trace output information about the managesdk task being performed.
- **-verbose**: Can be added to any getter task (-getXXX, -listXXX). For each SDK configuration setting retrieved by the task, the corresponding SDK metadata will also be displayed.
- **-enableServers**: Can be added to the -enableProfile and -enableProfileAll tasks. When a profile is enabled, the SDK setting for the profile's node is changed to the specified SDK name. The -enableServers parameter is a switch that, when present, is used to clear the SDK settings for all the profile's application servers that might have been explicitly set through another interface. This ensures that the application servers will now inherit the SDK setting of their parent node. Without it, all application servers SDK settings will remain unaltered.

The documentation contains complete details for using the managesdk command, including its use with federated profiles and security considerations.

The ManagedObjectMetadata commands and APIs

WebSphere Application Server V8.5 starts caching information about the newly installed SDKs as part of the node metadata properties for a node when one of these trigger actions is performed on a node after the SDK is installed:

- Starting a server. The type of server can be a deployment management server, an admin agent server, a job manager server, a node agent server, or a standalone application server.
- Creating a profile. This action results in a new node.
- Running the managesdk -enableProfileAll or managesdk –enableProfile command.
- Running the AdminTask.setNodeDefaultSDK() or AdminTask.setServerSDK() command.

The information collected about an installed SDK includes its version, installed location, bits, installed platform, installed architecture, and native libpath for the SDK installation.

Without performing one of those trigger actions, the information about a newly installed SDK won't be added into the node metadata properties. As a result, the ManagedObjectMetadata getter commands, the ManagedObjectMetatata getter APIs, and admin console “Java SDK” panels won't show any information about the newly installed SDK for a node or server.

Once the SDK information is collected as part of the node metadata properties for a node, you can query managed object metadata for SDK information through the wsadmin AdminTask commands or the WebSphere Application Server APIs. You can also view the available SDKs via the Java SDK administrative console panel for each node or server.

Two wsadmin AdminTask commands and a set of WebSphere Application Server APIs can be used to get the SDK information that is available on a given node. They are:

- **getAvailableSDKsOnNode**: Run the command to return a list of the names of the installed SDKs for a given node.
• **getSDKPropertiesOnNode**: Run the command to return a list of the properties of the SDKs installed for a given node.

See the **WebSphere Application Server V8.5 Information Center** for more details on these commands.

**ManagedObjectHelper APIs:**

- public `java.lang.String[] getAvailableSDKsOnNode(java.lang.String nodeName)` throws AdminException
- public `java.util.Properties getSDKPropertiesOnNode(java.lang.String nodeName, java.lang.String sdkName, java.lang.String[] sdkAttributes)` throws AdminException
- public `java.lang.String getSDKLocation(java.lang.String nodeName, java.lang.String sdkName)` throws AdminException
- public `java.lang.String getSDKVersion(java.lang.String nodeName, java.lang.String sdkName)` throws AdminException
- public `java.lang.String getSDKBits(java.lang.String nodeName, java.lang.String sdkName)` throws AdminException

These APIs belong to the ManagedObjectMetadataHelper class, which is part of the com.ibm.websphere.mangement.metadata package in WebSphere Application Server APIs. See the **WebSphere Application Server V8.5 Information Center** for more details on these APIs.

**The AdminSDKCmds group**

The AdminSDKCmds group in WebSphere Application Server V8.5 provides multiple AdminTask commands that can be used to manage (get, set, or clear) the SDK configuration for a node, server, or cluster runtime. These AdminTask commands are:

- **getNodeDefaultSDK**: Returns the values of the default SDK for a node, which includes the Java home and SDK name.
- **getSDKVersion**: Returns the version number of the SDK in use for a node, a server, or a cluster.
- **getServerSDK**: Returns the values of the SDK for a server. If a valid SDK value is set for the server, the returned values include the Java home and SDK name of the default SDK for the server.
- **getUnusedSDKsOnNode**: Returns a list of the names of SDK that a node is not using.
- **setNodeDefaultSDK**: Assigns a default SDK for a given node using specified the SDK Java home or the SDK name, but not both.
- **setServerSDK**: Assigns an SDK for a given server or cluster using the specified SDK Java home or the SDK name, but not both. The command can be also used to clear the server SDK setting by not specifying a value for SDK Java home or the SDK name.

See the **WebSphere Application Server V8.5 Information Center** for more details on those AdminTask commands.
Tips

- **Enable profile scripts and tools to use the same SDK enabled for the profile's server runtime**
  By default, when an application server is created, it contains no SDK configuration settings, and so inherits them from its parent node. However, the AdminTask setServerSDK() can be used to explicitly set a server's SDK configuration settings. When this situation exists, the profile's client scripts and tools could be running a different SDK than the server's runtime environment. This can lead to inconsistencies and run time exceptions. The profile's client scripts and tools, as well as the node and application servers, should all use the same SDK settings.

  If the profile's server SDK setting is different from the profile's client scripts and tools SDK settings, they can be made to be the same with the use of the managesdk command task -enableProfile along with the parameter -enableServers. This parameter clears out all the SDK configuration settings for the profile's application servers so they once again inherit the SDK configuration settings of the parent node. Listing 41 shows and example.

  **Listing 41**

  ```
  > cd <WAS_INSTALL_ROOT>\bin
  > managesdk.bat -listEnabledProfileAll
  CWSDK1004I: Profile default :
  CWSDK1006I: PROFILE_COMMAND_SDK = 1.6_32
  CWSDK1008I: Node MYSERVER_default SDK name: 1.6_32
  CWSDK1009I: Server server1 SDK name: 1.7_32
  CWSDK1011I: Successfully performed the requested managesdk task.
  > managesdk.bat -enableProfile
  -profileName default
  -sdkName 1.6_32
  -enableServers
  CWSDK1017I: Profile default now enabled to use SDK 1.6_32.
  CWSDK1001I: Successfully performed the requested managesdk task.
  > managesdk.bat -listEnabledProfile -profileName default
  CWSDK1004I: Profile default :
  CWSDK1006I: PROFILE_COMMAND_SDK = 1.6_32
  CWSDK1008I: Node MYSERVER_default SDK name: 1.6_32
  CWSDK1009I: Server server1 SDK name: 1.6_32
  CWSDK1011I: Successfully performed the requested managesdk task.
  ```

- **Make no dependencies on the format of an SDK name**
  As mentioned earlier, even though all currently supported SDK names follow the pattern of `<version>_<bitness>`, there is no formal convention for supported SDK names. If a new SDK is added to an existing WebSphere Application Server release, or if one is added in a future release, the SDK name may or may not follow the `<version>_<bitness>` pattern that is currently used. No customer script or application should make any assumption about SDK names that might be supported in the future.

- **Disable use of optional SDKs before uninstalling them**
  SDK 6, which was used in the initial install of WebSphere Application Server V8.5, must not be uninstalled. Any other SDK that was optionally installed can be removed — but before it is uninstalled, you must ensure no commands or servers are enabled to use it.

  In the example shown in Listing 42, the SDK 7 named 1.7_32 is going being uninstalled.
Listing 42

```bash
> cd <WAS_INSTALL_ROOT>/bin
> managesdk.bat -listAvailable
CWSDK1003I: Available SDKs :
CWSDK1005I: SDK name: 1.6_32
CWSDK1005I: SDK name: 1.7_32
CWSDK1001I: Successfully performed the requested managesdk task.
> managesdk.bat -getNewProfileDefault
CWSDK1007I: New profile creation SDK name: 1.7_32
CWSDK1001I: Successfully performed the requested managesdk task.
> managesdk.bat -setNewProfileDefault -sdkName 1.6_32
CWSDK1022I: New profile creation will now use SDK name 1.6_32.
CWSDK1001I: Successfully performed the requested managesdk task.
> managesdk.bat -getCommandDefault
CWSDK1006I: COMMAND_DEFAULT_SDK = 1.7_32
CWSDK1001I: Successfully performed the requested managesdk task.
> managesdk -setCommandDefault -sdkName 1.6_32
CWSDK1021I: The command default SDK name is now set to 1.6_32.
CWSDK1001I: Successfully performed the requested managesdk task.
```

Get a list of all the profiles in the installation to determine the SDK configuration that is used by each (Listing 43).

**Listing 43**

```bash
> managesdk.bat -listEnabledProfileAll
CWSDK1004I: Profile default :
CWSDK1006I: PROFILE_COMMAND_SDK = 1.7_32
CWSDK1008I: Node MYSERVER_default SDK name: 1.7_32
CWSDK1009I: Server server1 SDK name: 1.7_32
CWSDK1001I: Successfully performed the requested managesdk task.
```

In the resulting list, for each profile that uses the SDK that will be uninstalled, use the steps covered in sections *Enable managed nodes/servers/clusters to use SDK 7* and *Enable standalone application server profiles and management profiles to use SDK 7* to use the SDK named 1.6_32 instead of 1.7_32.

**Restart server to make newly installed SDK available for a node to use**

After installing or uninstalling an SDK, restart all of the server types below to trigger updates to the node metadata properties for the SDKs that are available for use:

- Node agent server
- Standalone application server
- Management server (deployment manager server, admin agent server, or a job manager server, will trigger the newly installed/uninstalled SDK information to be added/removed in/from its node metadata properties)

**SDK setting scope**

An SDK setting at a server level takes precedence over an SDK setting at a node level.

**Application deployment considerations**

- Make sure that the SDK version with which the applications are compiled is not higher than the SDK version that their application server runtime is using.
- Avoid using deprecated APIs.
- **Enable a deployment manager to use the correct version and fix level of the SDK**
You can enable the deployment manager to run any SDK version, but if you need to perform code generation while deploying your application, then the deployment manager and the target node must be running the same version and fix level of the SDK, and the same version and fix level of the WebSphere Application Server.

Conclusion

With a pluggable SDK, you can optionally configure commands and server runtimes to use SDK 7. You can choose to enable a single server, a node, a cluster, a profile (which includes all servers and commands within the profile), or your entire WebSphere Application Server V8.5 installation to use SDK 7. Additionally, you can use a pluggable SDK to enable either 32-bit or 64-bit versions of SDK 7 on IBM System i, and to enable either 31-bit or 64-bit versions of SDK 7 on z/OS.
Related topics

- WebSphere Application Server V8.5 product information
- WebSphere Application Server V8.5 Information Center
- AdminSDK Cmds group
- Installing and uninstalling IBM WebSphere SDK Java Technology Edition Version 7.0 on distributed operating systems
- managesdk command
- WebSphere Application Server V8.5 trial version

© Copyright IBM Corporation 2012

Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)