ORACLE ENTERPRISE MANAGER 13C PLUGIN DEVELOPMENT

PRACTICAL TIPS FROM AN EXPERIENCED PLUGIN DEVELOPER

AN AIDEV WHITE PAPER | MAY 2020





Revision History

The following changes have been made to this document:

Date	Revision
01.04.2020	Initial draft
18.05.2020	First released version

Credits

A special thanks to Dan Koloski and Sumesh Balakrishnan at Oracle for their review of this paper.

About the Author	4
Overview	5
Assumptions	5
The Main Stages of Plugin Development	6
Stage 1 – Identify plugin requirements	7
Stage 2 – Source test environments	8
Stage 3 – Understand the technology	9
Stage 4 – Profile the required targets	10
Stage 5 – Identify and define target metrics	13
Stage 6 – Develop the backend code	20
Stage 7 – Stage the plugin	22
Stage 8 – Validate and compile the plugin	26
Stage 9 – Import the plugin	27
Stage 10 – Deploy to the OMS	28
Stage 11 – Deploy to the agent	28
Stage 12 – Add a custom target	29
Stage 13 – Custom UI development	35
Additional Considerations	40
Custom Reports	40
Custom Jobs	40
Target Discovery	41
Oracle Enterprise Manager Mobile Application	41
Conclusion	42
Further Information	42

About the Author

<u>AIDEV</u> is a UK-based independent Oracle Enterprise Manager developer operated by Wardrop Consulting Limited, an Oracle Partner company.

It currently has several plugins on the <u>Oracle Enterprise Manager Extensibility</u> <u>Exchange</u>:

- mongoDB
- NGINX
- SSL Certificate
- HBase
- Symantec VCS cluster
- REDIS Data Store

AIDEV has also engaged in the creation of custom plugins for Oracle Enterprise manager customers, developed to meet their specific requirements.

Custom plugins enable seamless integration of non-Oracle supplied target types into the Oracle Enterprise Manager 13c monitoring, alerting, reporting and configuration management frameworks. The plugins are particularly suited to existing Oracle Enterprise Manager customers who need to monitor/alert on new technology but lack the knowledge or experience to do so easily.

AIDEV has been leveraging the Oracle Enterprise Manager Extensibility Development Kit (EDK) to develop custom plugins since Oracle Enterprise Manager 10g.

By using the EDK's powerful plugin development features, AIDEV can deliver enterprise-grade plugins that provide customers with the functionality they require.

Overview

This document details each of the main stages of plugin development using the Oracle Enterprise Manager 13c Extensibility Development Kit (EDK).

Each stage in the process is explained, based on experience gained by plugin developer AIDEV.

The aim of this document is to explain how plugin developers can leverage the EDK to create powerful enterprise-grade custom plugins which seamlessly integrate into Oracle Enterprise Manager 13c.

Assumptions

This document assumes the reader has basic experience of Oracle Enterprise Manager 13c and the Extensibility Development Kit, including the components of a plugin.

The Main Stages of Plugin Development

Oracle Enterprise Manager 13c plugin development is a multi-stage process.

At AIDEV, we perform each of the following distinct stages when developing a new plugin:

- 1. Identify plugin requirements
- 2. Source test environments
- 3. Understand the technology
- 4. Profile the required targets
- 5. Identify and define target metrics
- 6. Develop the backend code
- 7. Stage the plugin
- 8. Validate and compile the plugin
- 9. Import the plugin
- 10. Deploy to the OMS
- 11. Deploy to the agent
- 12. Add a custom target
- 13. Custom UI development

Each stage identified above is described in detail within the following sections of this document.

By following this process, plugin developers can easily create enterprise-grade plugins for Oracle Enterprise Manager 13c.

Stage 1 – Identify plugin requirements

The first stage in the development process is to identify exact requirements for the new plugin.

For example, these could be a combination of any of the following:

- To monitor and alert on core component status across an environment
- To manage component configuration standards across a site
- To provide single-screen visibility of all components within a cluster
- To allow remote component control (stop/start) from within Oracle Enterprise Manager 13c
- To provide a reporting capability for application availability

If we are developing a custom plugin for a customer, this stage typically involves the plugin developer having meetings with the customer to identify and document all of the main plugin requirements.

The developer may also need to liaise further with technical experts or support staff to gain an understanding of how the target technology works.

Investigation should be performed into how monitoring can be achieved, taking into consideration any metric retrieval interfaces that are published by the technology.

Once the main requirements for the plugin have been identified, the development process can move on to the next stage – sourcing a suitable test environment.

Stage 2 – Source test environments

Obviously this is a very important stage – plugin developers clearly require a test build, sandpit, VM, physical or similar environment to connect to and develop the plugin against.

They will also need to deploy and test the plugin against each Oracle Enterprise Manager supported by the plugin.

Target Environment

It is essential that all major releases and configurations of the custom target(s) are catered for and can be tested against.

This is particularly important for plugins that intend to support multiple target versions. Our mongoDB plugin supports multiple versions of mongoDB (version 2.3 to the current release 4.x) hence requires test environments for each major mongoDB release.

We typically use lightweight virtual machines for test environments as they help facilitate speed of delivery whilst maintaining flexibility in environment build options.

Oracle Enterprise Manager Environment

Plugin development must not be performed against a Production Oracle Enterprise Manager environment. Plugins will have to be deployed multiple times as part of the testing process and failed deployments may require a backout/restoration of the whole Enterprise Manager environment.

It is also essential that all major versions of Oracle Enterprise Manager supported by the plugin are tested against. The EDK version used to create the plugin needs to support all Enterprise Manager versions being tested against. For example, the developer may choose to develop the plugin using EDK v13.3 and certify it against Enterprise Manager 13.3, 13.4.

We strongly recommend using dedicated Enterprise Manager environments for development, ideally hosted on virtual machines to facilitate easy backout/restoration.

Once the required test environments have been sourced, the plugin developer should begin to gain familiarity with the target technology. This is detailed in the next section 'Understand the technology'.

Stage 3 – Understand the technology

It is extremely important that the plugin developer has a technical understanding of the target technology prior to developing the plugin.

Expert level knowledge is not essential. A fundamental understanding of the technology will however help ensure the plugin is appropriate for purpose.

The developer should assess whether a local or remote Enterprise Manager agent should be used to monitor the target technology. Remote agents allow for easy monitoring of multiple targets running across varied hosts and operating systems, whereas local agents permit the capturing of metrics and running of OS jobs without the need for remote connectivity.

For example, the following steps were required when developing our mongoDB plugin:

- Learn the core configuration aspects of the technology.

Example: MongoDB uses 'sharding' - data distribution across multiple machines. An understanding of mongoDB sharding directly influenced our decision to maintain a single target type for mongod and mongos type instances. This allowed the gathering of common metrics for all instance types whilst providing additional sharding metrics for mongos instances.

- Understand the security model.

Example: Following extensive testing, we were able to identify the roles/privileges needed to retrieve metric information. This in turn defined the configuration required within mongoDB to allow Oracle Enterprise Manager monitoring.

- Investigate how targets can be monitored programmatically. Example: Various tests were performed against sandpit environments before we opted to use the mongoDB Java drivers for our development. This approach provides the greatest portability and flexibility for our code, whilst permitting remote metric capture and JavaScript based job execution.

If the developer has limited knowledge of the technology, they should aim to learn the required skills, using online materials and testing against sandpit environments.

They may also choose to engage with specialists to further their understanding of the technology.

Once the developer has a sufficient level of technical understanding, they should begin to assess and document the required target(s), as detailed in the next section.

Stage 4 – Profile the required targets

It is essential that a plugin developer understands the target(s) required for a new plugin. Target properties and relationships need to be identified and defined prior to commencing plugin development.

Target properties

Target properties can serve the following purposes:

- They identify each attribute of the custom target
- They provide information to the agent-side plugin code to enable connectivity to the target and perform metric retrieval
- They can permit association between related targets

A custom target will have multiple properties. For our mongoDB plugin, we identified the following properties for the *mongodb_db* target type:

Property Name	Optional	Read Only
servername	N	N
port	Ν	N
jarloc	Ν	Ν
ssl	Ν	Ν
sslstore	N	N
mongo_id	Y	Ν

These properties map directly to the inputs provided when manually adding a mongoDB target into Enterprise Manager:

roperties	
* Monitoring Host Jar File Location	/home/oracle/mongojars
* Port	27017
* SSL Truststore	NONE
* SSL enabled [TRUE FALSE]	FALSE
* Server Name	mgo011.aidev.uk
mongoDB Environment Identifier	PROD_HR

Security credentials

Optionally, security credentials may also be required to allow the backend code to communicate with the managed component.

Monitoring credentials normally map to target properties, initially supplied when the target is added.

Our mongoDB plugin uses the following properties to authenticate against a dedicated monitoring account to retrieve metric data:

Property Name	Optional	Read Only
Username	Ν	N
Password	Ν	N

These properties map directly to inputs provided when adding in new mongoDB targets and result in security credentials of type 'mgoCreds' being created in Enterprise Manager:

mongoDB Credentials	
Credential type	mgoCreds
* Username	em_monitor
* Password	
* Confirm Password	

Plugin developers must identify and document the required target properties/credentials for each target type.

Target relationships

Some plugins may require multiple related target types, whilst other plugins might contain only one target type.

Target Type	Relationship
vcs_cluster	Top level cluster target - can
	<pre>contain 1x vcs_node targets</pre>
vcs_node	Node level target - can contain
	1x vcs node group targets
	Member of vcs_cluster
vcs_node_group	VCS group target per node - can
	contain 1x vcs_resource targets
vcs_group	VCS group target- cluster type
	target containing 1x
	vcs node group targets
	Provided by vcs_cluster target
vcs_resource	Bottom level target - maps to
	cluster resource in VCS
	Member of vcs node group target

Our VCS plugin contains the following related target types:

Target relationships aid the use of core Enterprise Manager features such as Topology Viewer and problem root cause analysis.

If related target types are required, developers should identify and document all required target types and relationships during this stage.

Once the developer has identified and documented the required target properties, security credentials and relationships, this data will input into subsequent stages of the development process.

At this point, the developer should then move on to the next stage in the process – metric definition.

Stage 5 – Identify and define target metrics

A custom target type will require various metrics to determine component status, performance and configuration.

We find that at this stage it helps to identify and document the metrics that will be implemented, their data types and any meaningful thresholds that can serve as a basis for alerting.

Metrics can then be added into the required XML files within the plugin stage directory tree (see Stage 7, '*Stage the plugin'*). Each defined metric will also map to programmatic logic developed within the backend code (see Stage 6, '*Develop the backend code'*).

This stage should be performed in conjunction with the next two stages. New metrics should be identified, corresponding logic developed in the backend code to allow agent retrieval of the required metric data, and finally metric definition and collection behavior added to the staged plugin XML files.

Agent Side Metrics

Traditional Enterprise Manager metrics are collected by the management agent, then uploaded to the OMS. Most metrics within a plugin will be of this type.

The first metric to define, and the only one which is mandatory, is **Response**.

This metric is required for all non-cluster target types and determines one column, **<u>Status</u>**, indicating the current target availability.

For our mongoDB plugin, the Response metric is documented as follows:

Metric Name	Col1	Collection
		Frequency
Response	Status [NUMBER]	Every 1 min

The next step a plugin developer should take would be to develop backend code to retrieve the Response metric (see Stage 6).

XML content for this metric can then be added to the staged target metadata and collection files(Stage 7).

For our mongoDB plugin, the *metadata* XML entry for the <u>**Response**</u> metric is as follows:



Once the Response metric has been identified, the backend code developed and the XML content added, the plugin developer should move on through each additional metric in turn.

A target type will normally contain many additional metrics – these could be singlecolumn or multi-column, single-row or multi-row, depending on the underlying data.

In the case of our mongoDB plugin, one example of a multi-column additional metric is the serverStatus **memory** metric.

This metric captures two columns – <u>Attribute</u> (the key value) and <u>Value</u>.

Data is collected every 15 minutes by default.

This metric is defined as follows:

Metric Name	Col1	Col2	Collection Frequency
Memory	Attribute [STRING] <mark>key</mark>	Value [NUMBER]	Every 15 min

The resultant metadata XML to be fed into Stage 7 for this metric is:



Note how the 'command' property runs the backend code, mongo.sh, passing in a parameter 'memory' - this returns metric data in the following format:



The corresponding *collection* XML for this metric is:



Consideration should also be given to using advanced metric columns. These are based on existing and previous values collected by the agent and are particularly useful for calculating rate/delta based pseudo values.

15 ORACLE ENTERPRISE MANAGER 13C PLUGIN DEVELOPMENT PRACTICAL TIPS FROM AN EXPERIENCED PLUGIN DEVELOPER

We use an advanced metric column within our mongoDB plugin to calculate the delta value for **deletes** based on the difference between current and previous measurements.

This is the XML content within the target *metadata* file:



Metric alerting thresholds also need to be identified. For our mongoDB plugin, the replication lag metric should provide the ability to alert when the collected lag is greater than a given threshold.

This is the resultant *collection* XML:



Most metrics will only be collected when a target is up, however this is not always the case (for example, log file content metrics).

The following *collection* XML example from our VCS plugin illustrates a metric **GroupState** that is also collected when the target is down:

<CollectionItem NAME="GroupState" UPLOAD_ON_FETCH="TRUE" COLLECT_WHEN_DOWN="TRUE"> <Schedule> <IntervalSchedule INTERVAL="5" TIME_UNIT="Min"/> </Schedule> <Condition COLUMN_NAME="group_state" OPERATOR="EQ"/> </CollectionItem>

Repository Side Metrics

In addition to standard agent side metrics, repository-side metrics may also be required.

Repository side metrics capture data from the Oracle Enterprise Manager repository metric data by running custom SQL. They are particularly useful when multi-target aggregate data or summary data is required for a metric.

To calculate the total size of all databases within a mongoDB target, our plugin uses a repository-side metric running the following SQL:

```
select target_guid, total as TOT from
(
    select target_guid, sum(value) as total
    from mgmt$metric_current where metric_name = 'dbStats'
    and metric_column = 'dataSizeMb'
    group by target_guid
)
```

This translates into the following metadata XML for Stage 7:



Repository side metrics are rendered alongside agent side metric in an identical manner:



17 ORACLE ENTERPRISE MANAGER 13C PLUGIN DEVELOPMENT PRACTICAL TIPS FROM AN EXPERIENCED PLUGIN DEVELOPER

Configuration collection

Configuration collection metrics are another consideration - data collected from these metrics can be viewed in Oracle Enterprise Manager under the 'Configuration' menu option.

If a custom target type exposes configuration data, this can be defined as a configuration collection and collected regularly.

Configuration collection metrics need to be marked **CONFIG=TRUE** in the metadata xml and require a custom table to be defined within the plugin to hold the data.

In the case of our mongoDB plugin, we capture build configuration for each target on a daily basis.

Two columns, **<u>ATTRIB</u>** and **<u>SETTING</u>** are captured by the backend code and stored as configuration data in the Enterprise Manager repository:

Metric Name	Col1	Col2	Collection Frequency
MONGODB_BUILDINFO2	Attribute [STRING] <mark>key</mark>	Value [STRING]	Every 1 day

The target *metadata* xml file contains the following entry for this metric:

<metric config="TRUE" name="MONGODB_BUILDINF02" type="RAW"></metric>
<pre><uisplay></uisplay></pre>
<label nlsid="mgo_buldinto2">mongoDB Buld Into</label>
<tabledescriptor table_name="MONGODB_BUILDINFO2"> <columndescriptor column_name="ATTRIB" is_key="TRUE" name="ATTRIB" type="STRING"> <display></display></columndescriptor></tabledescriptor>
<pre><label nlsid="attr label">Attribute</label></pre>
(c)umphescriptor
<pre></pre>
<pre>vispidy / shall NISTD="cett label"\value/label\</pre>
<pre></pre>
<pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>
<pre><pre><pre><pre>competity NAME="scriptsblr" Scope="StSTEMGLOBAL">Scriptsblr</pre>/Property></pre></pre></pre>
<pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>
<property name="ENVMGD_HOST" scope="INSTANCE">servername</property>
<property name="ENVMGO_USERNAME" optional="TRUE" scope="INSTANCE">Username</property>
<property name="ENVMGO_PASSWORD" optional="TRUE" scope="INSTANCE">Password</property>
<property name="command" scope="GLOBAL">/bin/bash %scriptsDir%/mongo.sh buildinfo</property>
<property name="delimiter" scope="GLOBAL"> </property>
<property name="ENVMGO_JARLOC" scope="INSTANCE">jarloc</property>
<property name="ENVMG0_SSL" optional="TRUE" scope="INSTANCE">ssl</property>
<property name="ENVMG0_SSLSTORE" optional="TRUE" scope="INSTANCE">sslstore</property>
<credentialref name="monCreds">mongoCredsMonitoring</credentialref>

A matching entry in the *collection* xml file enforces daily collection of the metric:

<CollectionItem NAME="mongdb_buildinfo2_snap" UPLOAD_ON_FETCH="TRUE" CONFIG="TRUE" COLLECT_WHEN_DOWN="FALSE"> <Schedule OFFSET_TYPE="INCREMENTAL"> <IntervalSchedule INTERVAL="24" TIME_UNIT="Hr"/> </Schedule> <MetricColl NAME="MONGODB_BUILDINF02" /> </CollectionItem>

This corresponds to an entry in the *snapshotlive* xml file, detailing the custom table and UI labels to use:

<pre><metadata snap_type="mongdb_buildinfo2_snap" target_type="mongodb_db" ver="1.0"></metadata></pre>
<pre></pre>
<001/IMN NAME="ATTRIR" TYPE="STRING" TYPE FORMAT="1024" IS KEY="Y">Config Param
<pre></pre>

| |
| |
When the plugin is imported into Oracle Enterprise Manager, we get a custom table created for the configuration data:

SQL>desc MONGODB_BUILDINFO2		
Name	Null?	Туре
ECM SNAPSHOT ID	NOT NULL	RAW (16)
ATTRIB	NOT NULL	VARCHAR2 (1024)
SETTING	V.	ARCHAR2 (1024)

The table underpins the mongoDB configuration data rendered within Oracle Enterprise Manager:

Config Param	Value
maxBsonObjectSize	16777216
modules	0
ok	1.0
openssl.compiled	OpenSSL 1.0.1f 6 Jan 2014
openssl.running	OpenSSL 1.0.1f 6 Jan 2014
operationTime.\$timestamp.i	1
operationTime.\$timestamp.t	1575407765
storageEngines	["devnull", "ephemeralForTest", "mmapv1", "wiredTiger"]

Once a metric has been identified and defined, the plugin developer should move on to the next stage, backend code development.

Stage 6 – Develop the backend code

The next stage in the plugin development process, following metric definition, is to create the method of communication between the Oracle Enterprise Manager agent and the custom target – the *'backend code'*.

The primary focus of this stage is to create a communication and metric retrieval mechanism. This will ensure that the metric data is captured and returned to the agent in an appropriate format.

Oracle Enterprise Manager can monitor targets in a variety of ways, for example remote or local, through perl scripts, bash scripts or Java RMI. If the agent can communicate with the target to capture the required metrics and return the metric data in a way that Enterprise Manager can understand, the list is practically endless.

At AIDEV, we begin by creating standalone scripts, passing in appropriate values for expected target properties and evaluating the retrieved output.

As mentioned previously, the **Response** metric should be targeted first – this governs target status and needs to return a number, either '1' (up) or '0' (down), to the agent.

In this case of our mongoDB plugin, the captured metric data is prefixed with *'em_result='* - expected output from the backend code for an 'Up' target is:

em result=1

As each additional metric is defined, the backend code should be enhanced to capture the required data.

Multi-column metrics will require a returned payload separated by a delimiter– this is usually a pipe symbol.

As indicated earlier, for our mongoDB plugin the backend script output for the **memory** metric is in the following format:

virtual|1645 bits|64 mappedWithJournal|0 mapped|0 resident|30

Appropriate error handling is also required in the backend code to ensure error-free data is passed to the agent.

Within our mongoDB plugin java source code, we handle connection timeouts in the following manner, passing '0' back for the response metric if a timeout is encountered:



Each metric should map to logic within the backend code and be defined with a *command* property in the staged target *metadata* XML (see next section).

For example, in the case of our mongoDB plugin, the asserts metric call is:

```
<Property NAME="command"
SCOPE="GLOBAL">/bin/bash %scriptsDir%/mongo.sh
asserts</Property>
```

We have found that test harnesses help to exhaustively test each metric being called by the backend code. They also allow for output comparison when running against varied configurations and versions.

An example of this is the following script, used to test each mongoDB metric within our plugin:



Once the backend code has been developed for a specific metric, the developer should move to the next stage, '**Stage the plugin**', adding metric definition and collection XML into the required plugin files.

Stage 7 – Stage the plugin

The next stage in the process is to gather all information and scripts from the previous stages and bundle them together into a skeleton plugin.

The most straightforward approach to take is to develop metrics individually, performing stages 5, 6,7 and 8 each time.

Normally at this stage, we don't include any custom UI content – this is added later in the process.

A prerequisite to performing this stage is to install the Enterprise Manager Extensibility Development Kit (EDK) and configure the plugin staging directory structure and XML files.

It should be noted that recent versions of the EDK reflect the move from Adobe Flex to JET for UI development. For this reason, we recommend using the 13.2 EDK u170321 or later with patch 25453518 – this maintains plugin support for Enterprise Manager 13.2 and above whilst permitting a JET based UI.

All examples in this document are based on EDK 13.2 – we use this version for our mongoDB plugin to ensure compatibility with Enterprise Manager 13.2 and above.

EDK install & staging area creation

The sample plugin code contained within the Enterprise Manager EDK zip file is a good starting point to begin with when creating the staging area.

The EDK can be obtained from the following menu in Oracle Enterprise Manager:



Deployment

- Download the Extensibility Development Kit to your workstation.
- Set your JAVA_HOME environment variable and ensure that it is part of your PATH. You must be running Java
 - 1.7.0_111 or greater. For example:
 - setenv JAVA_HOME /usr/local/packages/j2sdk1.7.0_111
 - setenv PATH \$JAVA_HOME/bin:\$PATH
- Unzip the downloaded zip file to your local system. For example:
 - Unzip 13.2.0.0.0_edk_partner.zip
 - It will create a bin directory under the directory where you have unzipped
- Change to bin directory and run empdk help (for example "empdk -help") from command line, for more details on
- empdk verbs

Download the EDK from this page and follow the configuration instructions provided.

Create the plugin staging area by copying the location samples/plugins/HostSample/demo_hostsystem/opar/stage and amending the content as follows:

plugin.xml

- XML to detail plugin metadata
- edit and amend according to plugin definition

agent/plugin registry.xml

- XML to detail metadata and files within the plugin
- edit and amend according to plugin definition

agent/metadata/target type.xml

- the main target XML file detailing target properties and metrics
- add in target metric XML as identified in Stage 5

agent/default collection/target type.xml

- metric collection behavior
- add in collection XML as identified in Stage 5

agent/scripts

- location for each backend script developed in Stage 6
- copy each script into here

agent/discovery

 leave this location empty for now as target discovery is beyond the scope of this example oms/metadata/assoc oms/metadata/derivedAssocs oms/metadata/discovery oms/metadata/snapshotlive oms/metadata/systemStencil oms/metadata/systemUiIntegration

leave these locations empty as the features are beyond the scope of this example

oms/metadata/targetType/target_type.xml

- this is typically identical to the agent equivalent

oms/metadata/default collection/target type.xml

- this is typically identical to the agent equivalent

oms/metadata/mpcui/target type.xml

- this would typically hold the mpcui UI definition, page content and menu layout prior to implementing a JET UI
- as a starting point, create a minimum content xml file based on the supplied sample projects

Note: It is extremely important that the plugin developer defines the *AgentCompatibility* tag within the staged plugin.xml file.

This ensures that backward compatibility can be achieved between the newly deployed OMS-side plugin and older agent-side plugin versions.

This is important when upgrading to a newer plugin release.

For our mongoDB plugin version 13.2.0.1.0 file, we define the following:

<AgentSideCompatibility> <Version>12.1.0.9.0</Version> <Version>12.1.0.10.0</Version> </AgentSideCompatibility>

The above entry ensures that agent-side plugin versions 12.1.0.9.0 and 12.1.0.10.0 can work against the OMS-side 13.2.0.1.0 plugin.

Please refer <u>here</u> for further information on the required XML file content in each of the required files.

XML content creation

The Oracle Enterprise Manager EDK contains multiple sample projects. At AIDEV we recommend using these as a starting point for XML file creation and a point of reference.

Oracle also provides a java-based tool, Plug-in Builder, for generating plugin XML files. This can be particularly helpful for plugin developers with limited experience of plugin development.

We normally develop our plugins manually, using previous plugin content as a starting point. We do however recommend Plug-in Builder for first time plugin developers to gain familiarity with the process.



Plug-in Builder is outside the scope of this document, however more information can be found at <u>https://docs.oracle.com/cd/cloud-control-13.3/EMPRF/GUID-6A94EE77-</u>D7AA-4A30-83AA-B627C41D7264.htm#EMPRF12922.

As a developer creates new plugins, existing code can be reused from previous builds/plugins.

We recommend adding each metric individually, each time performing a plugin validation as detailed in the next stage to troubleshoot any issues.

Stage 8 – Validate and compile the plugin

The empdk utility is bundled within the Oracle Enterprise Manager EDK.

It is used to compile an opar file from the plugin source directory structure created in the previous stage.

The opar file can then be imported into Enterprise Manager and deployed to the OMS and agent tiers.

Empdk also allows validation of the plugin content, checking for errors and highlighting areas requiring further attention.

To run empdk validation, the command is:

```
empdk validate_plugin -stage_dir {stage location} -out_dir
{opar location} -debug {debug log location}/debug.log
```

Sample output from a successful validation would be:

validating	
Validating Plug-in metadata	Passed
Validating Plug-in metadata semantics	Passed
Validating Staging Directory	Passed
Validating MRS Syntax	Passed
Validating MRS MPCUI	Passed
Validating MRS Semantics	Passed
Validating Metadata embedded SQL	Skipped
Validating Object Names	Passed
Plugin validation Passed	
Validation Report generated to: /home/orgc]	e/nlugin-dev/13200 EDK/nlugins/mongo/ongrs/nlugin validation report 191122 txt

Stages 7 and 8 should be performed in small increments, adding new metrics, fixing issues and re-validating the plugin.

Once all metrics have been added and final validation is complete for a plugin, it can be compiled into an opar file:

empdk create_plugin -stage_dir {stage location} -out_dir {opar location} -debug {debug log location}/debug.log

Sample output:

Validating Plug-in metadata	Passed 200_CDK/plugins/nginx/opars -debug /nome/oracle/plugin-
Validating Plug-in metadata semantics	Passed
Validating Staging Directory	Passed 200 EDK/ programs / toginx / ueougaog
Validating MRS Syntax	Passed
Validating MRS MPCUI	Passed
Validating MRS Semantics	Passed
Validating Metadata embedded SQL	Skipped
Validating Object Names	Passed
Plugin validation Passed	
Validation Report generated to: /home/orac	le/plugin-dev/13200_EDK/plugins/nginx/opars/plugin_validation_report_191119.xml
Creating the opar file	
Successfully created the plugin archive .	The oper file is /home/oracle/plugin-dev/13200_EDK/plugins/nginx/opers/13.2.0.1.0_gidev.nginx.xngx_2000_0.oper

Important: The plugin version defined in the staged plugin.xml and agent/plugin_registry.xml files must match the version of EDK being used, and in turn the version of Enterprise Manager.

Any attempt to run empdk against a different version of plugin will result in the following error:

[oracle]\$ empdk create_plugin -stage_dir /home/oracle/plugin-dev/13200_EDK/plugins/mongo/plugin_dist_v132010 -out_dir /home/oracle/pl ugin-dev/13200_EDK/plugins/mongo/opars -debug /home/oracle/plugin-dev/13200_EDK/plugins/mongo/debug.log Validating Plug-in metadata .. Error: Plug-in version specified in plugin.xml must match exactly with the one specified in agent/plugin_registry.xml

Once the opar file has been created, it can be imported into Oracle Enterprise Manager and deployed for further testing.

Stage 9 – Import the plugin

In this stage, the developer will import the plugin opar file created in the previous section.

At AIDEV, we recommend using a non-Production Enterprise Manager system for plugin development – failures in plugin deployment can require full OMS or OMR restoration to resolve.

Virtual machine hosted environments allow for a quick reversal of plugin deployment hence should be considered for the Oracle Enterprise Manager system.

The plugin opar can be imported into Enterprise Manager in the standard way:

emcli import update -file={full path to opar file} -omslocal

This imports the plug-in into the Enterprise Manager environment and makes it visible within the console.

Example:

\$ emcli import_update file="/home/oracle/13.2.0.1.0_aidev.mongo.xdbs_2000_0.opar" -omslocal

Processing update: Plug-in - Aidev mongoDB system monitoring plugin for Oracle Enterprise Manager Successfully uploaded the update to Enterprise Manager. Use the Self Update Console to manage this update.

Once the plugin is imported into Enterprise Manager, it should be deployed to the OMS and agent tiers.

27 ORACLE ENTERPRISE MANAGER 13C PLUGIN DEVELOPMENT PRACTICAL TIPS FROM AN EXPERIENCED PLUGIN DEVELOPER

Stage 10 – Deploy to the OMS

Deploy the plugin to the OMS tier using the standard procedure.

Stage 11 – Deploy to the agent

Deploy the plugin to the management agent using the standard procedure.

At this point, the plugin should be visible within Oracle Enterprise Manager and showing as being deployed to a management agent:

	lanager Cloud C	ontrol 13c			🛃 Enterprise * 🎯 Targets * 🌟 🕐 * 🔅 * 🔍 🌲 SYSMAN *
Plug-ins This page lists the plug-ins available, down	loaded, and deploye	d to the Enterprise	Manager system. Us	e this page to depk	Page Pefreshed Mer 30, 2020 édé:16 AM EDT 4
Actions = View = 👘 Deploy On	* 👘 Undeploy Fr	om * 🦨 Cher	ck Updates 🔲 D	Deployment Activit	les
		Version		Managament	
Name	Latest Available	Latest Downloaded	On Management Server	Agent with Plug-in	Description
Applications ()					
🕨 🛅 Cloud 🕕					
a in Databases					
Rese Database	13.2.0.1.0	13.2.0.1.0	13.2.0.1.0	1	Aidev HBase system monitoring plugin for Oracle Enterprise Manager
Cracle Database	13.4.1.0.0	13.4.1.0.0	13.4.1.0.0	0	Enterprise Manager for Oracle Database provides comprehensive management for Oracle Database and related targets such as Real Application Cluster, Automatic Storage Management (ASM) etc.
C mongoDB Database	13,2.0.1.0	13.2.0.1.0	13.2.0.1.0	1	Aldev mangoDB system manitoring plugin for Oracle Enterprise Manager
Engineered Systems ()					
🕨 🛅 Middleware 🕚					
Eng Servers, Storage and Network					
Cracle Audit Vault and Detaba	13.4.1.0.0	13.4.1.0.0 *		0	Enterprise Manager for Oracle Audit Vault and Database Firewall (AVDF) provides monitoring and management of AVDF system.
Cracle Beacon	13,4.0.0.0	13.4.0.0.0	13.4.0.0.0	1	Oracle Beacon plugin is required on the Managed Hosts to support beacon test monitoring capability
Cracle Consolidation Planning	13.4.1.0.0	13.4.1.0.0 °		N/A	Enterprise Manager for Oracle Consolidation Planning and Chargeback provides metering, chargeback and consolidation planning for various Enterprise Manager targets.
Cracle ORAchk Healthchecks	13.4.1.0.0	13.4.1.0.0 %		N/A	Enterprise Manager for Oracle ORAchk Health Checks provides proactive health check alerts for Engineered and Non- Engineered Systems.

The plugin developer should now proceed to the next stage, adding custom targets into Enterprise Manager.

Stage 12 – Add a custom target

Once the plugin has been successfully deployed to the OMS and agent tiers, a custom target can be added into Enterprise Manager.

The following example illustrates the adding of a custom mongoDB target:



Setup->Add Target-> Add Targets Manually

Choose 'Add Non-Host Targets Using Declarative Process':



Click the spyglass:

Add Target Declaratively	>
Host	2
Targot Type	
Target Type	
ADF Business Components for Jav	8
ASM IO Server	
ASM Proxy	
Access Manager - Access Server	
Access Manager - Identity Server	
Activity Graph	
Analytics	
Apache HTTP Server	
Automatic Storage Management	
-	
	Add Cancel

Select the management agent host to add the target to, then click Select



Select the custom target type in the target type box:



Click Add to view the Target Properties screen

The target properties screen will be shown:

	nterprise Manager Cloud Control 13c
Add: mongoDB Da	atabase
Add a target to be monitored	by Enterprise Manager by specifying target monitoring properties
Target	
* Target Name	MONGODB_1
Target Type	mongoDB Database
Host	emcc.aidev.uk
Agent	https://emcc.aidev.uk:3872/emd/main/
mongoDB Credentia	Is
Credential type	mgoCreds
* Username	em_monitor
* Password	
* Confirm Password	
Properties	
* Monitoring Host Jar File Location	/home/oracle/mongojars
* Port	27031
* SSL Truststore	NONE
* SSL enabled [TRUE FALSE]	FALSE
* Server Name	mongo1.aidev.uk
mongoDB Environment	MONGO_PRODUCTION

Complete the required properties and click OK

The target will be added into Enterprise Manager:



The target should now be visible and 'Up' in Enterprise Manager:

ORACLE' Enterprise Ma	nager Cloud Control 13c	Enterprise * () Jargets *	★* ©* ‡*	🔍 🌲 🛛 SYSMAN	* **
All Targets			Auto Refresh Off + Page P	efreshed Mar 20, 2020 6:11:04 Al	M EDT 🕤
Refine Search	View * Search Target Name mongodb			Save Search Saved Sear	ches =
🦼 Target Type	Target Name	**	Target Type	Target On F Status H	Remote Host
Databases mongoDB Database (1)	MONGODB_1		mongoDB Database	Ŷ	

31 ORACLE ENTERPRISE MANAGER 13C PLUGIN DEVELOPMENT PRACTICAL TIPS FROM AN EXPERIENCED PLUGIN DEVELOPER

On the target home page, the 'All Metrics' link will show the metrics defined for the target type. The following example illustrates the mongoDB memory metric defined earlier in this document:

MONGOD	B_1 🛈			
mongoDB 🔻				
NGODB_1 > All	Metrics			
Il Metrics				
earch	٩ 5	erver Sta	tus: Memory	
View +	11 12	Collection	Schedule Every 15 Minu	ites 🖋
Server S	tatus: GiobalLock	Uploa	ad Interval Every Collecti	on
Server S	itatus: LocalTime	La	ast Upload Mar 20, 2020	6:20:18 AM ED
Server S	tatus: Locks		Attribute	Value
 Server S Server S 	itatus: Locks Status: Memory	*	Attribute virtual	Value 1,753
Server S Server S Server S	tatus: Locks Status: Memory tatus: Metrics: queryEx	>	Attribute virtual	Value 1,753 64
Server S Server S Server S Server S	tatus: Locks Status: Memory tatus: Metrics: queryEx tatus: Metrics: record tatus: Metrics: Comman) 	Attribute virtual bits	Value 1,753 64
Server S	tatus: Locks Status: Memory tatus: Metrics: queryEx tatus: Metrics: record tatus: Metrics: Comman tatus: Metrics: Cureor	> > >	Attribute virtual bits mapped	Value 1,753 64 0
Server S	tatus: Locks Status: Memory tatus: Metrics: queryEx tatus: Metrics: record tatus: Metrics: Comman tatus: Metrics: Corsor tatus: Metrics: Docume r		Attribute virtual bits mapped mappedWithJo	Value 1,753 64 0
 Server S 	tatus: Locks Status: Memory tatus: Metrics: queryEx tatus: Metrics: record tatus: Metrics: Comma tatus: Metrics: Cursor tatus: Metrics: Docume tatus: Metrics: Operatic	+ + +	Attribute virtual bits mapped mappedWithJo	Value 1,753 64 0 0 298

After a period of time, the user can examine historical metric information for the target. The following example shows historical data for the mongoDB memory metric:

mongoDB = mongoDB =	1:17:37 PM EDT
Immunous * Very Data Last 24 Hours. All Metrics Very Technologic Mark 4, and Search Very Data Last 24 Hours. Very Technologic Mark 4, and Very Data Last 24 Hours. Search Very Data Addr Befreich Search Very Data Last 24 Hours. Search Very Data Addr Befreich Search Very Data Last 24 Hours. Search Very Data Addr Befreich Search Very Very Addr Befreich Addr Befreich Search Status Addr Befreich Addr Befreich Search Status Addr Befreich Addr Befreich Search Status Search Status Metricule resident Search Status Metricule resident Search Status Search Status Metricule resident Search Status Search Status Metric Value History	1117IB7 PM EDT
All Metrics Search Vew w Coursent Search	
All Metrics List 24 Hours List 24 Hours List 24 Hours List 24 Hours List 24 Hours List 24 Hours List 24 Hours List 24 Hours List 24 Hours List 24 Hours List 24 Hours List 24 Ho	
Search Verw Attribute Attr	Off \$
Verw Attribute Attribute<	
 Becurdy: Users Berner Status: Kasts Gerner Status: Matrics: gampEr Gener Status: Matrics: Comman Gener Status: Matrics: Comman	
 Sover Status: Assets Sover Status: Record Factor Provided in the science of the scien	7:22 PM
 Server Status: Reduct conditions Server Status: Conditions Server Status: Metrics: Cond	7:22 PM
 Berner Status: Connections Berner Status: Metrics: countrals Berner Status:	7:22 PM
 Sarver Status: Localine Sarver Status: Metrice: Comma Sarver Status: Met	
 Sarver Status: Metrice: Comman Sarver St	holds
Banver Status: Merice: Course Server Status: Merice: Course S	
> Server Status: Metrice: Cooline Collection Timestamp Mar 21, 2020 1/57.22 PM EDT Comparison Operator > > Server Status: Metrice: coordina A Metric Value History Occurrencies Before Alert 1 Low / High Value 197 / 305 Corrective Actions None > Server Status: Metrice: Coordina A Metric Value History Options = Compare Keys Compare Keys <td< td=""><td></td></td<>	
J Sarver Status: Metrice: record 300 Server Status: Metrice: counter 300	
Value Low / High Value 197 / 305 Corrective Actions None > Server Status: Metric: course Image: Compare keys	
 Server Status: Metrice: cover Scatus: Metrice: cover Scatus: Metrice: Coveres Server Status: Metrice: Serveres Server Status: Metrice: Serveres	
 Server Status: Metrics: Comman Server Status: Metrics: Comman Server Status: Metrics: Docume Server Status: Metrics: Docume Server Status: Metrics: Comman Server Status: Metrics: Server Status: Server S	
Server Status: Metrics: Contrast Server Status: Metrics: Docume Server Status: Metrics: Departs Server Status: Metrics: Departs Server Status: Metrics: getLast	argets
Server Status: Metrics: Cuesor Sonore Stat	
Server Status: Metrics: Dearatic Server Status: Metrics: Dearatic Server Status: Metrics: Dearatic Server Status: Metrics: getLast Server Status: Metrics: getLast Server Status: Metrics: 150 Server Status: Metrics: 150	a* 1
> Server Status: Metrice: Operator 259 V V V V V V V V V V V V V V V V V V V	×
Server Status: Metrice: Repical 200 Server Status: Metrice: getLast Solution: V V V	
Server Status: Melroice getLast 150 Server Status: Network 150	
 Barrer Status, Network Barrer Manuel Off 100 	
h George Station Old	
P Denvir Denue, ON	
Server Status: Opcounters 50	
Fighter Status: OpcountersRept 0	
In Service Status: PHD Image: PHD 04 05 08 10 12 AM 02 04 05 08 10 12 PM Minicro 120200 Minicro 120200 01 12 AM 02 04 05 08 10 12 PM	
Server Status: Process Value: resident	

Metric collection behavior can be amended through the 'Metric and Collection Settings' link.

The following example illustrates the mongoDB plugin memory metric default collection settings:

ORACLE	Enter	prise Manager Cloud C	ontrol 13c			Enterprise *	Targets *	*	•	ф.	9		SYSMAN	
* MONGODB_	10												8.	ncc.sidev.ik
mongoDB Database: MON Edit Collection Set	NGCOB_1 : ttings: S	 Metric and Collection Setting erver Status: Memory 	Edit Collection Section Sec	ettings: Server Statu	is: Memory									
Editing the collection settin	ngs of a me	tric will also affect the collection	settings of other metr	rics that are based or	n those metrics. The affe	cted metrics are listed in th	e Affected Metrics secti	ion.					Cancel	Continue
Collection Schedule	· · · · ·													
Data Collection Enabled	Disable													
Collection Frequency														
Default Frequency Every	30 Minute	í												
Prequency Type	8	y Minutes 🕈												
Repeat Every	15	Minutes												
Use of Metric Data														
Alerting and Historical Uplcad Interval 1 C Grow an alert is detected, a Alerting Only Alert history will be saved.	I Trending Collections data will be a	deaded to the repository menufately.				The Upload Interval deter collected every 5 minutes	mines how often a me , and the Upload Interv	tric value is u val is set to 6	ploaded to th (every 6th c	he Manageme offection), the	nt Repositor metric valu	y. For exa e is upload	mple, if a metric Sed every 30 mir	value is utes
Affected Metrics														
Affected Metrics -														1
Server Status: Memory														
Server Status: Memory	of all metri	cs that belong to the above liste	d metric groups will be	e affected.										Cance

Custom key-based alerting thresholds can also be added. In this example, we have added Warning and Critical thresholds for the value obtained for the *virtual* key:

MONGODB_1 MongODB * mongoDB testimase: K0NCODB_1 > Metric and Collection Settings > Edit Advanced Settings: Value Edit Advanced Settings: Value Alert Message Edit Alert Message Red Alert Message The value of %ecclumeNameNameNameNameNameNameNameNameNameNa	Alert Message Prope ese properties can be use imme	orties	erroc aldex.uk
mongoDB + mongoDB + mongoDB tatabase: K0X600B_1 > Maric and Collection Settings > Edit Advanced Settings: Value Edit Advanced Settings: Value Alert Message Edit Art Message The value of "AccultureVientify for "AveryValue%s is "Nonlurfs" The Value of The To level in the page message around the most than 6000 characters. The Value of The To level in the page message around the most than 6000 characters. The Value of The Value of the page message around the most than 6000 characters. The Value of The Value of the page message around the most than 6000 characters. The Value of The Value of the page message around the most than 6000 characters.	Alert Message Prope ese properties can be us	erties	Cancel Continue
	Alert Message Prope ese properties can be usi lame	ortics	Cancel Continue
Alert Message Edit Alert Message Edit Alert Message Recit Alert Message Alert Message The value of "AccolumcName%s for "AlertYValue%s is "funduate%s" The value of "AccolumcName%s for "AlertYValue%s"	Alert Message Prope ese properties can be usi lame	erties	
Edit Alert Message Edit Alert Message The value of Neuclannehitanehis far Neuclannehitanehis The Value of Neuclannehitanehis far Neuclannehitanehis The The Inseller of the about message compared to an operations The The Inseller of the about message compared to an operations	Alert Message Prope ese properties can be use lame	erties	
Alert Message The value of Nockum Namethy for NoveyValue's is Novalue's	ese properties can be use lame	erues	
TIP The length of the alert message cannot be more than 4000 characters.	ese properties can be us lame		
If TIP The length of the alert message cannot be more than 4000 characters.	lame	sed in message. Property names are case-sensitive. To escape '%', use '%%'.	
		Description	
	ometric_id%	Metric name for which the alert has been triggered	
9	icolumnName%	Metric column name for which the alert has been triggered	
٩	warning_threshold%	Threshold for which warning violation has been triggered.	
9	scritical_threshold%	Threshold for which critical alert has been triggered	
	bseverity%b	Severity level of the alert or violation	
	soperator%	Comparison operation used to trigger the alert	
	hnum_of_occur%	Number of Occurrences after which alert has been triggered	
2	walue?%	Current metric value on which alert has been triggered	
,	okeyValue%	Current metric value for a key on which alert has been triggered	
Monitored Objects			
The table lists all Attribute objects monitored for this metric. You can specify different threshold settings for each Attribute object.			Add Reorder
Edit Remove			
Select Attribute Comparison Operator Warning Threshold Critica	Threshold	Corrective Action	
# virtual > 2000 5000		None	
0 All others >		None	
TIP Fins (or Engly Thread-bids will disable about for that metric. TIP fins (or engly) and the stand theoretic to engenerat multiple objects. (Example: /s1% represents /s11, /s12 etc) TIP bit the object name contains "%" or "\", specify it as "\%" or "\" (Example: 'c)temp' needs to be entered as 'c-\temp')			

Once a target had been added, the plugin developer should test each metric from within Oracle Enterprise Manager.

Data collected and rendered within Enterprise Manager should be verified as being appropriate and correct.

The Metric and Collection Errors menu option can be used to identify issues:



Any issues or metric collection errors should be investigated, resolved, re-tested and the plugin source updated accordingly.

This action may involve additional tracing on the Enterprise Manager agent or by adding debug scripting into the backend code.

As the plugin code evolves, developers can use the '<u>emctl register oms</u> <u>metadata'</u> command to update Oracle Enterprise Manager with the changes made to core XML files.

Alternatively, if using a VM based development environment, Oracle Enterprise Manager can be quickly restored. Newly compiled opar files can then be imported, redeployed and subsequently tested again.

Following the successful undertaking of this stage, the plugin is nearing completion.

Developers should now progress to the next stage, custom UI development.

Stage 13 – Custom UI development

Once the plugin is imported into Enterprise Manager, and a target has been added, the plugin developer can create the custom plugin UI.

This is the final stage in the development process.

The plugin UI should be created using Oracle's <u>JavaScript Extension Toolkit (JET)</u> through the <u>Apache Netbeans</u> IDE.

Oracle has provided sample Netbeans projects within the EDK sample project code. We recommend using these as a starting point for UI development.

Apache Netbeans allows the plugin developer to create a JavaScript library that can be added into the plugin staging area and included within the final compiled opar file.

Once imported to the OMS, this library is used by Enterprise Manager to render the UI pages for target types contained within the custom plugin.

The Netbeans project contains code for each custom UI page – a JavaScript controller and a corresponding HTML based view page.

All controller code is contained within the js branch of the Netbeans project:



All screen content is defined in HTML files, within the View branch of the project:



Netbeans allows the plugin developer to iteratively code and test in a standalone IDE, independent of Oracle Enterprise Manager.

Plugin interfaces can be run in a standard web browser locally, allowing quick development and testing of each UI page.

We typically develop UI screens using Netbeans and the Google Chrome browser.

Chrome also provides a <u>Netbeans connector</u> – this allows for deeper tracing and debugging of the UI from within Netbeans.

The logon screen within the Netbeans test harness allows for custom UI code to be run against a specific target in Enterprise Manager:

Protocol	🔘 http 오 https
Management Server Host	192.168.0.77
Management Server Port	7803
Administrator Username	sysman
Administrator Password	•••••
Target Name	MONGODB_1
Target Type	mongodb_db

36 ORACLE ENTERPRISE MANAGER 13C PLUGIN DEVELOPMENT PRACTICAL TIPS FROM AN EXPERIENCED PLUGIN DEVELOPER

The following example is of the mongoDB target home page for the target added earlier in this document:



Metric information can be retrieved from Enterprise Manager and rendered graphically within the UI pages.

This screen illustrates the values obtained for the *memory* metric defined earlier in this document (all data on this screen is historical and retrieved directly from the Oracle Management Repository):

locks								
network bytes in [deita] bytes out [deita] num requests [deita]	serv	erStatus: i	memory			-	_	×
opcounters insert [delta] query [delta] update [delta]	Last	50 Minutes		٣				
delete [delta] getmore [delta] opcountersRepl	1.8K	-						
insert (delta) query [delta] update [delta] delete [delta] getmore [delta]	1.2K 0.9K						 Value (bits) Value (mapped) Value (mappedWith Value (mappedWith	thJournal)
repl	0.6K						 Value (resident) Value (virtual) 	
storageEngine	0.3K							
wiredTiger	0.0							
writeBacksQueued	0.0	10:15 AM	10:25 AM	10:35 AM	10:45 AM	10:55 AM		
memory [bits] [mapped] [mapped]	v	Hidi 20 2020						
metrics commands cursor (timedOut) [open.noTimeout] document [updated] [deleted] [res getLastError [wtime.num] [wtime.tot	itamaa, staimaa,	[wtimeouts]	_	_	_		_	Close

Screens can also capture and render real-time metric data, obtained directly from the custom target by the Enterprise Manager agent.

This example	e shows	currently	connected	l clients,	captured	real-time	from mongoDB:

MONGODB_1										📑 empolaidev.uk
mongoDB *								Page	Refreshed Fri Mar	20 2020 11:13:19 GMT+0000 (Greenwich Mean Time) 🗘
Clients										
This table details all cur	rent operations within d	b.currentOp								
Description	Client	Opid	Threadid	connid	Namespace	Active	Waiting	Op	numYields	Query
rsSync-0_56613		56613				*	false	none	0	0
manitaring keys for HMAC_5		5				~	false	none	0	0
conn36_56602	192.168.0.53:54471	56602		36	local oplog.rs	*	false	getmore	2	("\$repiData":1,"\$readPreference":("mode":"secondary9 ["t":1884702783;");); signature";("keyid":878617853 ("base64";"KTXQ1qX8088tBhU5tq.ePY5QgY7I=";"subT ("t":606;"ts":("\$timestamp":("t":1584702783;":";))); "ge
conn1943_56614	192.168.0.77-17162	56614		1943	admin.\$cmd.aggregate	*	false	command	0	("Ibid":("id':("\$binary":("base84":"fUsXcE5PkbTYn01T ("mode":"primaryPreferred"),"\$db":"admin","\$clusterTii ("keyld")0, "hash"("\$binary":("base64":"AAAAAAAA ("\$own0ps":"true"); \$atti:"true))
WT RecordStoreThread: local.oplog.rs_29		29				*	false	none	0	0
ReplBatcher_56612		56612				*	false	none	0	0

Traditional legacy Flex-type objects can be used within the UI pages – this is enabled through the inclusion of mpcui libraries within the supplied Netbeans project code.



Flex-type objects enable plugin developers with Flex experience to move to JET based coding with relative ease.

They also allow legacy Flex based plugins to be migrated to JET relatively quickly.

Oracle has documented the similarities between Flex and JET objects here.

For example, an availability data service in JET:

Compared to the Flex equivalent:

We recommend using Oracle's <u>documentation</u> as a starting point for UI development in JET.

The sample plugin project code contained in the EDK is also an excellent reference point for developers.

For the purpose of UI development testing, newly compiled JavaScript libraries can be imported directly into Oracle Enterprise Manager from a stage location:

```
emctl register oms metadata -service mpcui -file
/stage/mpcui/mongodb_database.xml -pluginId aidev.mongo.xdbs
```

This allows the developer to test UI changes from within Oracle Enterprise Manager.

Once UI development is complete, the resultant JavaScript libraries can be placed within the plugin code staging area.

The finished plugin opar file can then be compiled and imported into a clean Enterprise Manager environment for final testing.

Additional Considerations

Outside the scope of this document, some other areas of consideration when developing a plugin are:

Custom Reports

Developers can include BI Publisher and Information Publisher reports in their plugin code. Reports can be easily developed within Enterprise Manager then copied to the plugin staging area before opar recompilation.



Custom Jobs

Job definitions can be contained within a plugin's content.

Our mongoDB plugin facilitates component control as well as providing the ability to remotely execute JavaScript code against targets. These operations can be run seamlessly within the Oracle Enterprise Manager jobs framework.



Target Discovery

Plugins can use guided target discovery. Our VCS plugin allows for target discovery, initiating custom code to identify VCS components on a host, passing the relevant target and association information back to Enterprise Manager.

Symantec VCS Target Discovery Discovery	ery						
Add VCS Targets.	0	0					
Tips:	Select Agents	Configure Inputs	Configure Targets Sum	mary			
Select Agents: Select the agents you wish to add VCS targets to - typically this should include all cluster hosts. Agents must have the VCS management agent placin designed	Add VCS Ta	argets: Confi	gure Targets			Step 3 of 4	Next Cancel
Seforehand. Click Next to proceed to the Configure Inputs stage.		Select Targ	ets To Add to Console				
		Status	Agent	Target Name	Target Type	Set Proper	
Configure Inputs: If required, provide a naming tag for the associated VCS targets in the		Unmanaged	vcs2.localdomain:3879	ClusterService.cluster1.CLUS1	vcs_group	Properti	
This may be used to auto generate target names to ensure unique namina across the EM estate.		Unmanaged	vcs1.localdomain:3879	ClusterService.cluster1.CLUS1	vcs_group	Properti	
To add targets to an existing EM managed cluster, ensure that this is set, correctly,		Unmanaged	vcs1.localdomain:3879	ClusterService.vcs1.cluster1.C	vas_node_group	Properti	
This may be left blank if cluster names are unique across your estate.		Unmanaged	vcs2.localdomain:3879	ClusterService.vcs2.cluster1.C	vcs_node_group	Properti	
Confirm whether a privilege access tool is to be used. This influences the use of a tool, eg. sudo, by both the target discovery		Unmanaged	vcs1.localdomain:3879	ORCL-iP.oraGroup.vcs1.cluste	vos_resource	Properti	
process and target metric collection. Click Next to proceed to target discovery.		Unmanaged	vcs2.localdomain:3879	ORCL-IP.oraGroup.vcs2.cluste	VC6_F660UFD6	Properti	
		Unmanaged	vcs1.localdomain:3879	ORCL-Netisnr.oraGroup.vcs1.a	VC6_resource	Properti	
Target discovery results are displayed. Click the Properties button to view detected target properties.		Unmanaged	vcs2.localdomain:3879	ORCL-Netisnr.oraGroup.vcs2.4	vcs_resource	Properti	
Served the fargets you wish to add into EM and click Next.		Unmahaged	vcs1.locaidomain:3879	ORCL-vol.oraGroup.vcs1.clust	vcs_resource	Properti	
		Unmanaged	vos2.iocaldomain:3879	ORCL-vol.oraGroup.vcs2.clust	VOS_MISOUTOR	Properti	

More information on the above features can be found within the <u>Extensibility</u> <u>Programmer's Guide</u>.

Oracle Enterprise Manager Mobile Application

Oracle Enterprise Manager 13.4 introduces a new <u>mobile application</u> – this allows the EM administrator to seamlessly interact with the EM monitoring and incident management framework through an Android/Apple app.

Plugins do not require any additional functionality to leverage this capability – integration is provided out of the box, allowing remote monitoring and incident management of custom targets through a mobile device.

Targets			Incident		+	Ta	rget				
Filters (1) Clear 10 Results	Target Status 17	Target Status 17 DETAILS EVENTS UPDATES				★ mgo_r1_0 ★ mongoDB Database					
mgo_r1_1		The value of St	atus is O		OPEN INC	IDENTS					
 mongoDB Database 		\$	<u>.</u>	New	FATAL	CRITICAL	WARNING	ESCALATED			
mgo_r1_0		FATAL	UNASSIGNED	STATUS	• 0	0 🖸	<u>^</u> 0	0			
 mongope parapase 		PRIORITY	UNACKNOWLEDGED	ESCALATED	ppopppt	50					
mgo_r1_2					PROPERT	55					
Inongobo balabase		TARGET			Up Since						
mgo_r0_1		- map =0 1					Apr 22, 2020 11:01:28 AM				
E confidence annuale		Ingo_ro_r mongo	DB Database		100	1.403					
mgo_r0_2					Agent						
		INCIDENT DETA	ILS		Time Zoo	W.UK.3072					
mongoDB Database		1D			Eastern D	aylight Time					
man confinally		2223									
mongoD8 Database		4/22/20 3:46:54	PM								
man confindb?		Last Updated									
mongoDB Database		4/22/20 3:47:17	PM								
- mao r0 0		ADDITIONAL INF	FORMATION								
🔹 🐨 mongoDB Database											
	100 C	Category									

41 ORACLE ENTERPRISE MANAGER 13C PLUGIN DEVELOPMENT PRACTICAL TIPS FROM AN EXPERIENCED PLUGIN DEVELOPER

Conclusion

At AIDEV we have found that the Oracle Enterprise Manager 13c EDK enables rapid and powerful development of custom plugins.

By following the stages outlined in this document, plugin developers can create enterprise-grade plugins to meet custom requirements.

Plugins created in this manner allow for a seamless integration into Oracle Enterprise Manager 13c, leveraging full use of core features such as monitoring, alerting, reporting and configuration management.

Apache Netbeans allows for iterative development of feature-rich JET based screens, rendering core target metric and configuration information to the end-user.

The inclusion of mpcui libraries within the EDK sample projects reduces the learning curve experienced by Flex developers and allows for a smooth migration of existing Flex-based plugins to a JET/js/HTML format.

Further Information

Aidev http://www.aidev.uk

Oracle Enterprise Manager 13c https://www.oracle.com/technetwork/oem/enterprise-manager/overview/index.html

Oracle Enterprise Manager Extensibility Exchange <u>https://apex.oracle.com/pls/apex/oracle_enterprise_manager/r/em-extensibility-exchange-v3</u>

Oracle Enterprise Manager 13c Extensibility Documentation https://docs.oracle.com/cd/cloud-control-13.3/nav/extensibility.htm

AIDEV/Wardrop Consulting Ltd is an Oracle Partner based in the UK, specializing in Oracle Enterprise Manager.

We offer Oracle Enterprise Manager consultancy services, including custom plugin development for application vendors.

Please contact info@aidev.uk for more information.

