

# Receiving SNMP Traps in Oracle Enterprise Manager

*An Oracle White Paper*  
*June 2009*

# Receiving SNMP Traps in Oracle Enterprise Manager

Introduction.....	3
Common Usage Scenarios.....	4
Receiving SNMP Traps.....	4
Design the Management Plug-in.....	5
Map the SNMP Traps to OEM alerts.....	5
Develop the metadata files that make up a Management Plug-in.....	7
Validate the metadata files.....	16
Create the Management Plug-in archive.....	16
Deploy the Management Plug-in.....	16
Add a new target in OEM.....	17
Update the trap destination.....	17
Optional: add a monitoring rule for new target type.....	18
Conclusion.....	19

# Receiving SNMP Traps in Oracle Enterprise Manager

## Introduction

Oracle Enterprise Manager (OEM) provides top-down application management for Oracle technologies including Oracle packaged applications, Oracle Fusion Middleware, Oracle Database and Oracle VM. However, it is often critical for an administrator to receive alerts from applications that are not managed by OEM. Many of these applications can be configured to trigger SNMP traps when an alert condition takes place. You can receive these traps within OEM and start monitoring those applications from OEM. Since OEM is optimized for analyzing historical alerts, having the capability to receive and analyze SNMP traps raised by such applications allows you extend OEM's monitoring and alerting capabilities to these applications and reduce monitoring complexity in your IT environments.

This paper provides a step-by-step guide for receiving SNMP traps in Oracle Enterprise Manager.

**Note:** The example used in this paper is based on the SNMP framework provided by the Oracle Contact Center Anywhere (CCA) application. You may use the general guidelines specified here to build a plug-in to receive SNMP traps from any application. If you want to send SNMP traps from OEM out to a trap receiver, refer to the SNMP support reference guide.<sup>1</sup>

This document complements the official OEM configuration<sup>2</sup> and the Official EM Extensibility guide<sup>3</sup>.

---

<sup>1</sup> SNMP Support Reference Guide

[http://download.oracle.com/docs/cd/B16240\\_01/doc/em.102/b16244/toc.htm](http://download.oracle.com/docs/cd/B16240_01/doc/em.102/b16244/toc.htm)

<sup>2</sup> Advanced Configuration Guide:

[http://download.oracle.com/docs/cd/B16240\\_01/doc/em.102/e10954/toc.htm](http://download.oracle.com/docs/cd/B16240_01/doc/em.102/e10954/toc.htm)

<sup>3</sup> Extensibility Guide:

[http://download.oracle.com/docs/cd/B16240\\_01/doc/em.102/b40007/toc.htm](http://download.oracle.com/docs/cd/B16240_01/doc/em.102/b40007/toc.htm)

## Common Usage Scenarios

SNMP traps can be received in OEM from any application in your IT environment.

Examples of scenarios when one would want to configure OEM to receive SNMP traps include:

Steps to receive SNMP traps:

1. Design a management plug-in
2. Deploy the management plug-in
3. Add a target to OEM
4. Update the trap destination
5. Optional: Create notification rules

- *Application server fronted by a load balancer:* The middleware administrator may want to be alerted when the load-balancer that fronts his application server is facing performance problems. OEM can be configured to listen for load balancer traps.
- *Oracle packaged application using a critical network switch:* OEM can be configured to listen for SNMP traps raised by critical network switches or routers.
- *OEM as the central monitoring console:* OEM can be configured to receive SNMP traps from all SNMP enabled applications in the IT environment. Thereby making it the central console of event monitoring in the datacenter.

## Receiving SNMP Traps

We want to configure OEM to receive the SNMP traps raised by an application (figure 1) and display the traps as alerts in the OEM console.

```
version[1] messageId[0] SnmpSubAgent : 19 : ***** : Receive : new message (1) from HostManag
1] version[1] messageId[0]
: 0 : 7293856 : 24046 : SnmpSubAgent : 19 : ***** : Got : DISCONNECTED from HostManager
: 5 : 7293856 : 24046 : SnmpSubAgent : 19 : ***** : Checking for last resource type 10, group
: 5 : 7293856 : 24046 : SnmpSubAgent : 19 : ***** : Queuing : new message (17) from HostManag
61] version[1] messageId[0]
: 5 : 7293856 : 24046 : SnmpSubAgent : 19 : ***** : Queuing : new message (1) from HostManag
1] version[1] messageId[0]
: 4 : -152181056 : 24046 : SnmpSubAgent : 19 : ***** : Get : new message (17) from HostManag
1] version[1] messageId[0]
: 4 : -152181056 : 24046 : SnmpSubAgent : 19 : Receive : New [C] message, message
: 4 : -152181056 : 24046 : SnmpSubAgent : 19 : ***** : Get : new message (1) from HostManager
] Id[1] group[-1062680561] version[1] messageId[0]
: 4 : -152181056 : 24046 : SnmpSubAgent : 19 : Receive : New [C] message, message
: 0 : -152181056 : 24046 : SnmpSubAgent : 19 : Stamp : Calling TWCSNMPSubAgent::AnyDisconnectH
andler
: 5 : -152181056 : 24046 : SnmpSubAgent : 19 : remove resource [1] from resource map
: 5 : -152181056 : 24046 : SnmpSubAgent : 19 : SOCKET_DISCONNECT is real [1]
: 0 : -152181056 : 24046 : SnmpSubAgent : 19 : entering SnmpSendTrap() with trap specific number [
:8] and extra info [ [HostManager=1]]
: 0 : -152181056 : 24046 : SnmpSubAgent : 19 : sent the trap [resourceStopped] with info []
: 4 : -152181056 : 24046 : SnmpSubAgent : 19 : sendQueryToStatsServer [insert into trapshistory (
id, resourceid, companyid, description, isdeleted) values ( 19001461468426, '1239916595', 8, 1, -1
r=1', 0 )] type[44000]
: 4 : -152181056 : 24046 : SnmpSubAgent : 19 : ***** : Sent : new message (44000) to StatsSer
on[1] ver Id[0] group[1] version[1]
: 0 : -152181056 : 24046 : SnmpSubAgent : 19 : Stamp : Leaving TWCSNMPSubAgent::AnyDisconnectH
andler
```

Figure 1: Showing the SNMP traps raised by CCA application

In order to receive these traps, we will develop a SNMP Receivelet based Management Plug-in. At a high level, you need to:

- Design the Management Plug-in
- Deploy the Management Plug-in
- Add a target to OEM
- Update the trap destination

- Optional: Create notification rule and subscribe administrators to notification rules

The first step involves designing a management plug-in. The plug-in makes OEM agent capable of receiving SNMP traps. The plug-in also tells OEM how the traps should be displayed on its console.

## Design the Management Plug-in

In order to design a management plug-in, you need to:

- Map the SNMP traps to OEM alerts.
- Develop metadata files that make up a Management Plug-in.
- Validate the metadata files.
- Create the Management Plug-in Archive.

### Map the SNMP Traps to OEM alerts

As the first step, we need to understand how the SNMP traps map to OEM metric alerts. For that, you need to obtain the traps from the CCA MIB. Find below an extract of a few of the traps from the CCA MIB<sup>4</sup>.

```
SNMPAgentShutdown TRAP-TYPE
ENTERPRISE telephonyatwork
VARIABLES { trapInfo }
DESCRIPTION "SNMPAgent is shutting down"
 ::= 1
.
.
SNMPAgentRunning TRAP-TYPE
ENTERPRISE telephonyatwork
VARIABLES { trapInfo }
DESCRIPTION "SNMPAgent is up and running"
 ::= 1001
.
.
companyDeleted TRAP-TYPE
ENTERPRISE telephonyatwork
VARIABLES { trapInfo }
DESCRIPTION "Company has been deleted"
 := 5
```

Notice that all traps use a single variable, trapInfo, shown in the **Table 1**.

An individual SNMP trap maps to one and only one OEM metric while a single OEM metric can map to one, two or three SNMP traps. For example, a single OEM metric can be created to process trap ID 1 and 1001 above. We will map trap ID 1 to an OEM alert with severity Critical and clear that alert on receipt of trap id 1001. Traps 5, 6, and 27 on the other hand, map to one and only one critical OEM

<sup>4</sup> To learn more about SNMP V1 Trap structure, go to [http://www.tcpipguide.com/free/t\\_SNMPVersion1SNMPv1MessageFormat-3.htm](http://www.tcpipguide.com/free/t_SNMPVersion1SNMPv1MessageFormat-3.htm)

alert metric since there is no corresponding Clear event trap ID defined. **Table 1** shows the mapping of some of the CCA Traps to OEM alert metrics.

SNMP Trap			Matching OEM Alert Metric
Trap ID	Name	EVENT	Name
1	SNMPAgentShutdown	Critical	SNMPAgentShutdown
1001	SNMPAgentRunning	Clear	
2	licenseFailure	Critical	licenseFailure
1002	licenseSuccess	Clear	
3	systemOverflow	Critical	systemOverflow
1003	systemLicenseOK	Clear	
29	sipDialOutFailure	Critical	sipDialOutFailure
1029	sipDialOutSuccess	Clear	
5	companyDeleted	Critical	companyDeleted
6	resourceCrashed	Critical	resourceCrashed
27	maliciousCalltrace	Critical	maliciousCalltrace

**Table 1.** Mapping of sample CCA Traps to OEM alert metrics

#### Construct OIDs from CCA MIB

Process the CCA MIB either by reading the file or with the help of any available MIB browser tool (there are several freeware versions available on the Internet). The table below lists some of the OIDs constructed from the CCA MIB.

Name	OID	Comment
Telephonyatwork	1.3.6.1.4.1.10477	Trap Enterprise
ReleaseVersion	1.3.6.1.4.1.10477.1.1	Scalar Attribute (Product)
AboutString	1.3.6.1.4.1.10477.1.2	
numberOfInteractions	1.3.6.1.4.1.10477.1.3	
CompanyTable	1.3.6.1.4.1.10477.1.4	Attributes of Company Table
CompanyEntry	1.3.6.1.4.1.10477.1.4.1	
CompanyIndex	1.3.6.1.4.1.10477.1.4.1.1	
CompanyId	1.3.6.1.4.1.10477.1.4.1.2	
CompanyAlias	1.3.6.1.4.1.10477.1.4.1.3	
companyInteractions	1.3.6.1.4.1.10477.1.4.1.4	
companyAgentLoggedIn	1.3.6.1.4.1.10477.1.4.1.5	
companyACDCall	1.3.6.1.4.1.10477.1.4.1.6	
CompanyACDChat	1.3.6.1.4.1.10477.1.4.1.7	
companyACDCallBack	1.3.6.1.4.1.10477.1.4.1.8	
companyACDWebCallBack	1.3.6.1.4.1.10477.1.4.1.9	
companyACDPredictive	1.3.6.1.4.1.10477.1.4.1.10	
companyACDEmail	1.3.6.1.4.1.10477.1.4.1.11	
companyACDFax	1.3.6.1.4.1.10477.1.4.1.12	

Name	OID	Comment
companyACDVoiceMail	1.3.6.1.4.1.10477.1.4.1.13	
companyAvailableAgents	1.3.6.1.4.1.10477.1.4.1.14	
companyACDSMSs	1.3.6.1.4.1.10477.1.4.1.15	
CompanyACDOutboundCalls	1.3.6.1.4.1.10477.1.4.1.16	
TrapInfo	1.3.6.1.4.1.10477.1.6	Scalar Trap Attribute

**Table 2.** CCA OIDs

The OIDs for Telephonyatwork and TrapInfo (**Table 2**) are OIDs needed for traps while the rest of the OIDs in Table 2 are accessible and can be queried as needed by the SNMP fetchlet.

For the purpose of this paper, our focus will be on traps. Nonetheless, reference will be made to these other accessible objects simply to illustrate that not all OIDs are necessarily SNMP trap related.

#### Identify Traps

Table 3 shows the alert traps and the clearing traps.

Alert Trap (Critical)		Clearing Trap	
Trap ID	Name	Trap ID	Name
1	SNMPAgentShutdown	1001	SNMPAgentRunning
2	LicenseFailure	1002	licenseSuccess
3	SystemOverflow	1003	systemLicenseOK
5	CompanyDeleted	N/A	
6	Resourcecrashed	N/A	
27	MaliciousCalltrace	N/A	

**Table 3.** Sample CCA Traps

Once we have identified the SNMP trap components, we can map them to EM alerts using the Management Plug-in metadata files. Table 4 summarizes the metadata files that constitute a Management Plug-in.

Name	Description	Comment
<target type>_ttd.xml	Target Type Metadata Definition	Deployed to \$OH/sysman/admin/metadata
<target type>_dc.xml	Target Default Collection Definition	Deployed to \$OH/sysman/admin/default_collection

**Table 4.** Plug-in Metadata Files

#### Develop the Metadata Files that Make Up a Management Plug-in

The first metadata file, the target type metadata definition (or ttd) XML file, defines the target type, the metrics and the methods the OEM agent will be using to collect those metrics. It contains the following required components:

- Document Type Definition (DTD) location

The DTD referenced here is the document that describes the content of the XML file and set constraints and restrictions on its structure. e.g.

```
<!DOCTYPE TargetMetadata SYSTEM "../dtds/TargetMetadata.dtd">
```

- META\_VER

This is the version of the metadata. Each time a plug-in is released with changes, the META\_VER should be bumped up.

- TYPE

The type uniquely identifies the kind of target instance this metadata file is for. The type must be unique within all target types of OEM.

- Display

The Display value is used in the OEM UI for the particular target type.

```
<TargetMetadata META_VER="1.0" TYPE="oracle_cca"
<Display>
  <Label NLSID="oracle_cca">Oracle CCA Application</Label>
</Display>
```

- Metrics

They are the dominant part of the metadata file and define all metrics to be monitored for the target type. Display strings, data columns and the method used to obtain the data values are all part of a metric definition.

**Note:** In order for a target type to be considered valid, it must have a metric called *Response* containing a metric column named *Status*. This metric is used to monitor the state of the target (UP/DOWN).

- QueryDescriptor

In the example shown below, the Response metric is obtained by using the OEM agent's SNMP fetchlet, which is specified in the QueryDescriptor. The QueryDescriptor also contains properties that are needed by a fetchlet. There are many different types of fetchlets supported by the OEM agent and SNMP fetchlet is a special type of fetchlet. The SNMP fetchlet queries accessible SNMP objects as defined in the target type's MIB. The TableDescriptor contains definitions of all of the metric columns (in this case, there is just the single Status column). Similar metric definitions could be defined with the metadata file so that the OEM agent could collect other information about the target instance that is exposed through the SNMP MIB.

If you are interested in creating metrics that have SNMP datapoints (vs. SNMP traps), refer to the OEM Extensibility Guide on OTN.

A **fetchlet** is a parameterized data access mechanism that takes arguments (For example, a script, a SQL statement, a target instance's properties) as input and returns formatted data. A fetchlet 'fetches' or retrieves information by querying at certain intervals. Each fetchlet handles a specific type of data access.

```

<Metric NAME="Response" TYPE="TABLE">
  <Display>
    <Label NLSID="oracle_cca_resp">Response</Label>
  </Display>
  <TableDescriptor>
    <ColumnDescriptor NAME="Status" TYPE="NUMBER">
      <Display>
        <Label NLSID="cca_SNMP_resp_status">CCA SNMP SubAgent
Status</Label>
      </Display>
    </ColumnDescriptor>
  </TableDescriptor>
  <QueryDescriptor FETCHLET_ID="Snmp">
    <Property NAME="NAME" SCOPE="INSTANCE">NAME</Property>
    <Property NAME="hostname" SCOPE="INSTANCE">AgentHost</Property>
    <Property NAME="PINGMODE" SCOPE="GLOBAL">TRUE</Property>
    <Property NAME="OIDS"
SCOPE="GLOBAL">1.3.6.1.4.1.10477.1.1.0</Property>
    <Property NAME="COMMUNITY" SCOPE="INSTANCE"
OPTIONAL="TRUE">CommunityString</Property>
    <Property NAME="TIMEOUT" SCOPE="INSTANCE"
OPTIONAL="TRUE">Timeout</Property>
  </QueryDescriptor>
</Metric>

```

The 'NAME' property contains the name of the CCA target instance.

The 'hostname' property contains the location of the SNMP agent.

The 'PINGMODE' property is typically used with the SNMP fetchlet within a Response metric. When it is set to TRUE, this property tells the fetchlet to return the SNMP target's availability status to a column defined in the Response metric.

The 'OIDS' property can contain a list of OIDs that are defined in the target type's MIB that correspond to properties that can be obtained. In this case, 1.3.6.1.4.1.10477.1.1 corresponds to the *releaseVersion* property in the MIB. The final 0 in the OID signifies that a scalar object (single value) will be returned. If there isn't a trailing 0, then a table column (multiple values) will be returned. In the example with the Response metric above, the OID specified is actually inconsequential since PINGMODE is set to TRUE.

The 'COMMUNITY' property is the SNMP community value. If not specified the default value is public.

The 'TIMEOUT' property corresponds to the SNMP timeout value, which defaults to 5 seconds.

- PushDescriptor

An OEM agent can also listen for data through the use of a *receivelet*. The OEM agent supports different types of receivelets. The SNMP receivelet can be used by the OEM agent to listen for SNMP traps as defined in the target type's MIB. A *PushDescriptor* is part of the metric definition of a receivelet.

A *Receivelet* is a library that allows Enterprise Manager to receive notifications sent by third-party network elements. A receivelet listens at a particular port for the notifications. These are notifications that are asynchronously sent and without any requests from Oracle Management Agent.

```
TELEPHONYATWORK-MIB DEFINITIONS ::= BEGIN
IMPORTS
    enterprises, gauge
        FROM RFC1155-SMI
    OBJECT-TYPE
        FROM RFC-1212
    DisplayString
        FROM RFC-1213;

telephonyatwork OBJECT IDENTIFIER ::= { enterprises 10477 }
serverPart OBJECT IDENTIFIER ::= { telephonyatwork 1 }
...
trapInfo OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION "Trap Description"
::= { serverPart 6 }
...
companyDeleted TRAP-TYPE
ENTERPRISE telephonyatwork
VARIABLES { trapInfo }
DESCRIPTION "Company has been deleted"
::= 5

::= 5
```

The type of receivelet is identified by the properties that the receivelet needs in order to listen for a particular SNMP trap. Find below a portion of CCA's MIB used in this example:

```
generic-trap – generic trap type
    INTEGER {
        coldStart(0),
        warmStart(1),
        linkDown(2),
        linkUp(3),
        authenticationFailure(4),
        egpNeighborLoss(5),
        enterpriseSpecific(6)
    }
```

In addition, there are standard SNMP MIB definitions that will be referenced as well. Below is the definition for generic-trap (RFC1157): The example below defines a metric, companyDeleted, which will listen for SNMP trap of the same name:

```

<Metric NAME="companyDeleted" TYPE="TABLE">
  <Display>
    <Label NLSID="companyDeleted">Company Deleted</Label>
  </Display>
  <TableDescriptor>
    <ColumnDescriptor NAME="trapInfoCompDel" STATELESS_ALERTS="TRUE"
TYPE="STRING" IS_KEY="TRUE">

  <Display>

<Label NLSID="TrapInfo5">Company Deleted Trap(Stateless)</Label>

</Display>
  </ColumnDescriptor>
</TableDescriptor>
  <PushDescriptor RECVLET_ID="SNMPTrap">
    <Property NAME="MatchEnterprise"
SCOPE="GLOBAL">1.3.6.1.4.1.10477</Property>
    <Property NAME="MatchGenericTrap" SCOPE="GLOBAL">6</Property>
    <Property NAME="MatchSpecificTrap" SCOPE="GLOBAL">5</Property>
    <Property NAME="MatchAgentAddr"
SCOPE="INSTANCE">AgentAddr</Property>
    <Property NAME="EventtrapInfoCompDelOID"
SCOPE="GLOBAL">1.3.6.1.4.1.10477.1.6</Property>
    <Property NAME="SeverityCode"
SCOPE="GLOBAL">CRITICAL</Property>
  </PushDescriptor>
</Metric>

```

Since the companyDeleted SNMP trap does not have a corresponding clearing SNMP trap, STATELESS\_ALERTS is set to TRUE. In the case, the OEM UI allows the user to manually clear the alert. When an SNMP trap has a corresponding clearing SNMP trap (i.e. systemOverflow and systemLicenseOK from Table 1), STATELESS\_ALERTS is set to FALSE.

The properties that can be used by the SNMP receivelet are defined in the following table:

Parameter	Type	Description	Use
MatchEnterprise	String	OID used to define the trap being sent.	Required
MatchGenericTrap	String	Code for a generic SNMP trap.	Required
MatchSpecificTrap	String	Trap defined in a MIB (not one of the generic traps), the ID assigned to that MIB.	Required
MatchAgentAddr	String	IP address of the generating SNMP agent, as sent in the trap.	Required
Event<metric-column>	String	Specifies that, on receiving this trap, the receivelet should generate a severity on this metric column. The value of the metric column should be the value of this parameter. (This case is useful where the expected	Required, if events have to be generated. However, if Event<metric-column>OID is

Parameter	Type	Description	Use
		values of the EM metric are not the same as the triggering SNMP variable.)	provided, then this is not required.
Event<metric-column>OID	String	Specifies that, on receiving this trap, the receivelet should generate a severity on this metric column. The value of the metric column should be taken from the varbind in the trap with OID equal to the value of this parameter.	Required, if events have to be generated. However, if Event<metric-column>is provided, then this is not required.
SeverityCode	String	Specifies the level at which the severity should be generated. The value of this parameter should be one of 'CRITICAL', 'WARNING' or 'CLEAR'.	Required. However, if SeverityCodeOID is provided, then this is not required.
SeverityCodeOID	String	Specifies the level at which the severity should be generated. If the varbind in the trap with OID equal to the value of this parameter is one of the strings 'CRITICAL', 'WARNING' or 'CLEAR', the severity should be generated at that level; otherwise, no severity should be generated. (This parameter would only be used if the integrator were designing a trap exclusively for use with EM, but may be useful in this case.	Required. However, if SeverityCode is provided, then this is not required.
Data<metric-column>OID	String	Specifies that, on receiving this trap, the receivelet should generate a datapoint on the metric, for which the value of this metric column should be taken from the varbind in the trap with OID equal to the value of this parameter. (An SNMP PushDescriptor may have many Data* parameters, in which case a single row will be returned, with all specified columns populated from the appropriate varbind in the trap. An SNMP PushDescriptor may not have both a Data* parameter and a Severity* parameter, nor may it have multiple Severity* parameters.)	Required, if datapoints have to be generated.
Key<metric-column>OID	String	Severity or datapoint generated by this PushDescriptor should contain a key value for this metric column. The key value should be taken from the varbind in the trap with OID equal to the value of this parameter. For every key column in the metric definition, there must be a Key* parameter in the PushDescriptor.	Optional
Context<metric-column>OID	String	If the PushDescriptor generates a severity, the severity should contain a value for this metric column in its event context. The value should be taken from the varbind in the trap with OID equal to the value of this parameter.	Optional. This can be used only for severities.

Parameter	Type	Description	Use
		If the PushDescriptor generated a datapoint, this parameter is ignored.	

**Table 5. SNMP Receivelet Properties**

In the companyDeleted metric example above, MatchEnterprise is set to telephonyatwork (1.3.6.1.4.1.10477), which is the highest-level object identifier based off of the standard enterprises value (1.3.6.1.4.1).

MatchGenericTrap is set to 6, which is the enterpriseSpecific value for generic-trap found in RFC1157. This implies that a specific trap value will also be supplied.

MatchSpecificTrap is set to 5, which is the value of the companyDeleted TRAP-TYPE in the MIB.

MatchAgentAddr is set to the AgentAddr instance property value for the target instance.

Event<metric-column>OID turns into EventtrapInfoCompDelOID and has the value 1.3.6.1.4.1.10477.1.6, which is the OID of trapInfo. The alert generated will be for the trapInfoCompDel metric column.

Finally, SeverityCode is set to 'CRITICAL' so when this SNMP trap is caught, then a critical level alert will be generated.

NOTE: In the current 8.1.2 CCA build, the trapInfo variable is not sent with the generated traps. Instead, the trap contains a single variable with the OID set to 1.3.1.4.1.10477.6. In addition, the Trap Enterprise value from the trap is 1.3.1.4.1.10477.6. This value 1.3.1.4.1.10477.6 does not exist in the CCA MIB. This means for 8.1.2 CCA, the definition of the companyDeleted metric as shown above should be modified so that both MatchEnterprise and EventtrapInfoCompDelOID properties have the value 1.3.1.4.1.10477.6.

- Instance Properties

These properties are those that uniquely identify a particular instance of a target type. The oracle\_cca target type will have four target instance properties as illustrated below. The Labels that are specified are used in the EM UI on the page where a target instance can be manually added.

```

<InstanceProperties>
  <InstanceProperty NAME="AgentHost" CREDENTIAL="FALSE"
OPTIONAL="FALSE">
    <Display>
      <Label NLSID="oracle_cca SNMP_agent_host">Hostname (SNMP Agent )
</Label>
    </Display>
  </InstanceProperty>
  <InstanceProperty NAME="AgentAddr" CREDENTIAL="FALSE"
OPTIONAL="FALSE">
    <Display>
      <Label NLSID="oracle_cca SNMP_agent_addr">IP Address (SNMP Agent)</
Label>
    </Display>

```

```

</InstanceProperty>
<InstanceProperty NAME="CommunityString" CREDENTIAL="TRUE"
OPTIONAL="TRUE">
  <Display>
    <Label NLSID="oracle_cca_SNMP_community_string">SNMP Read
Community String (Default: public)</Label>
  </Display>
  public
</InstanceProperty>
<InstanceProperty NAME="Timeout" CREDENTIAL="FALSE"
OPTIONAL="TRUE">
  <Display>
    <Label NLSID="oracle_cca_SNMP_timeout">SNMP Timeout (Default: 30
seconds)</Label>
  </Display>
  30
</InstanceProperty>
</InstanceProperties>

```

Setting CREDENTIAL to TRUE results in the instance property getting obfuscated while you type it in OEM's Add Target page.

When an instance property is set to be OPTIONAL, you need not specify it on the Add Target page. A default value can be specified as can be seen for the instance properties CommunityString and Timeout above.

The second metadata file, the target default collection (or dc) XML file, defines the schedule and collection interval for each metric added to the metadata type definition. It's made up of the following

- Document Type Definition (DTD) location

The DTD referenced here is the document describing the content of the XML file and setting constraints and restrictions on its structure. e.g.

```
<!DOCTYPE TargetMetadata SYSTEM "../dtds/TargetCollection.dtd">
```

- Target Type reference

The Target Type reference matches the TYPE set in the target type metadata file.

```
<TargetCollection TYPE="oracle_cca">
```

- Collection Item

A collection for each metric that is defined in the target type metadata file. This can either be obtained on a given schedule with a specific interval or simply by listening. When dealing with SNMP, the collection definition largely depends on whether the metrics are obtained through a fetchlet or a receivelet.

*Note:* A collection item must be defined for the Response metric so that the availability of the target instance is known to OEM.

In the example below, a `CollectionItem` is set up for the Response metric. The OEM agent collects it every 5 minutes. A `Condition` is defined for the Status metric column so that when the value is 0, a critical alert will be generated. For the Response metric there is no need to define a warning level alert. However, for other metric columns, it is likely that when a critical alert threshold is set, a warning level threshold would be set as well. In addition, messages can be defined for display when the alert occurs or is cleared.

```
<CollectionItem NAME = "Response">
  <Schedule>
    <IntervalSchedule INTERVAL = "5" TIME_UNIT = "Min"/>
  </Schedule>
  <Condition COLUMN_NAME="Status"
    CRITICAL="0"
    WARNING="NotDefined"
    OPERATOR="EQ"
    OCCURRENCES="1"
    MESSAGE="%target% is unreachable or is down."
    MESSAGE-NLSID="oracle_cca_response_status"
    CLEAR_MESSAGE="%target% is up."
    CLEAR_MESSAGE-NLSID="oracle_cca_response_status_clear"/>
</CollectionItem>
```

Unlike the `CollectionItem` for a metric that is obtained by a `fetchlet`, no schedule or interval is needed for a metric that gets its data from a `receivelet`. However, the `PUSH` property for the `Condition` must be set to "TRUE". This indicates that the SNMP traps are being pushed by the SNMP agent rather than being pulled by the OEM agent on a given schedule. See the example shown below:

```
<CollectionItem NAME="companyDeleted">
  <Condition PUSH="TRUE"
    COLUMN="trapInfoCompDel"

    COLUMN_NAME="trapInfo1"
    CRITICAL="1"
    MESSAGE=" companyDeleted trap received for target %target%: %value%"
    MESSAGE-NLSID="oracle_cca_tranInfoCompDel"
  </Condition>
</CollectionItem>
```

Messages can contain placeholders, which will be filled in when the alert message is created. Some of these placeholders include:

- `%value%` - value of the metric (or column of metric)
- `%target%` - name of the target
- `%column_name%` - can include value columns as well as key columns
- `%critical_threshold%`: the critical threshold of the condition

The next two sections cover detailed steps for building and deploying the Management Plug-in.

### Validate the metadata files

- Download the EM PDK (empdk.jar)<sup>5</sup> to validate the XML files to a workstation that has a 10.1.x or 10.2.x OEM agent installed on it.
- Set the ORACLE\_HOME environment variable to the OEM agent's ORACLE\_HOME.
- Set the JAVA\_HOME environment variable to the proper location.
- Add JAVA\_HOME/bin to your PATH environment variable.
- Install the PDK as follows:

```
java -jar emPDK.jar client -install_dir=<pdk_installation_directory>
```

- Add <pdk\_installation\_directory>/pdk/bin to your PATH environment variable and then run the following for command usage and summary:

```
emcli help check_mp
```

- Check the syntax of the metric metadata and the default collection files:

```
emcli check_mp -metadata_file=oracle_cca_ttd.xml -  
collection_file=oracle_cca_dc.xml -err_level=error
```

If the files are well constructed, the program will return with no error message. Next you need to package all the metadata files into a portable management plug-in archive.

### Create the Management Plug-in Archive

- Assuming that you have the same environment set up as when the metadata files were validated, use EMCLI to package the plug-in:

```
emcli add_mp_to_mpa -mpa="oracle_cca.jar" -mp_version="1.0"  
-ttd="oracle_cca_ttd.xml" -dc="oracle_cca_dc.xml" -func_desc="CCA Management  
Plug-in to receive SNMP Traps"
```

- Copy the generated JAR to your local desktop.

**Tip:** Deploy the plug-in to several agents in one go to reduce plug-in roll out time in your datacenter.

### Deploy the Management Plug-in

<sup>5</sup> Oracle Enterprise Manager 10g Grid Control Management Plug-in Development Kit [http://www.oracle.com/technology/products/oem/extensions/management\\_plugin\\_sdk.html](http://www.oracle.com/technology/products/oem/extensions/management_plugin_sdk.html)

### **Import the Archive into OEM**

- Go to: Setup>Management Plug-ins.
- Click the Import button.
- Browse to the JAR location and select the file.
- Click the List Archive button.
- Select the listed CCA plug-in.
- Click the OK button to import.

### **Push the plug-in to Agent(s)**

- Identify the row for the Management Plug-in you just imported.
- Click the Deploy icon for the CCA Management Plug-in.
- Click the Add Agents button.
- In the Search and Select popup window, set Target Type to Agent in the LOV.
- Select the agent(s) where you would like the plug-in deployed.
- Click the Select button to dismiss the popup window.
- Hit the Next button and then the Finish button.

### **Add a New Target in OEM**

Even though the CCA Management Plug-in has been deployed in OEM, it is not yet ready to accept and display the SNMP traps. You have to create a target instance within OEM that can display alerts.

Follow the steps below to add the target:

- Go to Setup>Agents.
- Select the agent the Management Plug-in has been deployed to by clicking on its link.
- Select the target type (Oracle CCA Application) from the Add dropdown and click Go.
- On the target configuration page, fill in the target instance name (a suggested format could be <target-type>\_<hostname>).  
[e.g. oracle\\_cca\\_myhost](#)
- Fill in the Hostname, which is the location of where the SNMP daemon for CCA is running.  
[e.g. myhost.domain.com](#)
- Fill in the AgentAddr as the IP address of Hostname.
- Optionally fill in CommunityString, which defaults to public.
- Optionally fill in Timeout as the value for the SNMP timeout, which defaults to 30 seconds.
- Click the OK button to add the target.

Now that OEM is ready to receive the SNMP alerts, you must point the SNMP trap originator to the OEM agent where the Management Plug-in is deployed.

### Update the Trap Destination

In order to successfully receive traps from CCA, the trap destination must be set to point to the host and port where the OEM agent is running (3872).

- On the CCA server, edit the file “/etc/SNMP/SNMPd.conf” and specify the value of trapsink as:

`trapsink <hostname> public 38726`

- As root, restart the SNMPd daemon (master agent).

`/sbin/service SNMPd restart`

- As ccauser, restart the CCA SNMP subagent. This can be done through the CCA host manager client application by logging into it, right-clicking on “SNMP subagent” and then restart.

Now go back to OEM and navigate to the newly added target to verify that the SNMP traps are being received and displayed (**figure 2**)

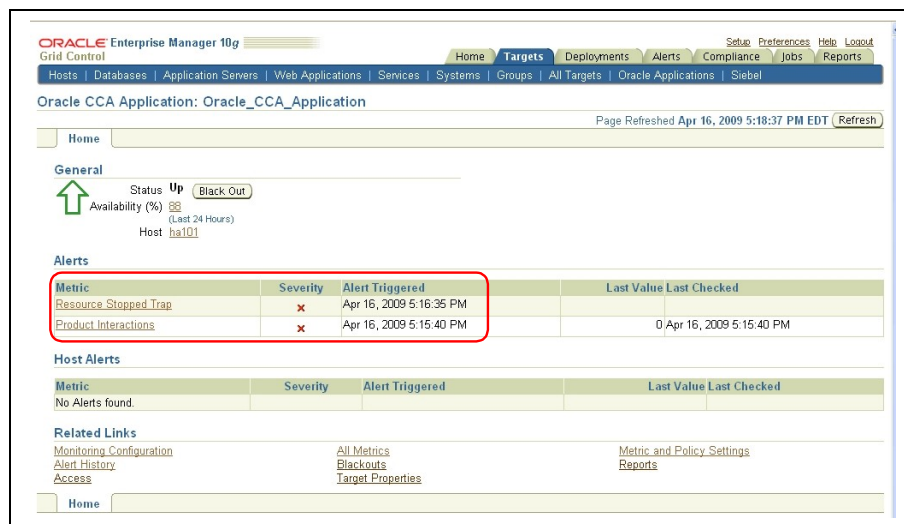


Figure 2: Showing SNMP alerts in the ‘Alerts’ section of EM console

### Optional: Add a monitoring rule for new target type

A notification rule tells EM what to do after an SNMP alert has been received. As an example, you may create a rule to send an email to an administrator when a particular SNMP trap comes in. The steps to write a notification rule are:

- Go to Preferences>Notification Rules.
- Click the Create button.
- Fill in the rule name.

### Oracle CCA Application Availability and Critical State

<sup>6</sup> The CCA Client should be installed prior to updating this file. Otherwise it will be overwritten.

- Fill in the Description.

#### [Custom rule for notifying on reception of Oracle CCA Application's SNMP Traps and monitoring Product and Company metrics](#)

- Select Oracle CCA Application from the Target Type LOV.
- Check Make Public to make it available to any administrator with access to such target type.
- On the Availability tab, check Up and Down to be notified when the target goes down or come up.
- On the Metrics tab, click the Add button.
- Select the metric and the severity state (critical, warning or clear) that you would like to be notified on and click the Continue button.
- On the Actions tab, specify if you want repeated email sent (the default is up to 3 with 15 minute intervals in-between) and whether you want yourself (i.e. the rule owner) also notified when an alert is issued.
- Click the OK button to submit.

#### **Subscribing Administrators to the Newly Added Monitoring Rule**

- Go to Setup>Administrators.
- Select the administrator (the selected administrator should already have an email address assigned to it) and click the Subscribe to Rules button.
- Select the newly added rule (Oracle CCA Application Availability and Critical State), and then click the Subscribe button.
- Click the OK button to save.
- Repeat these steps as needed for other administrators that need to be notified.

## Conclusion

Management Plug-ins provides a versatile mechanism for you to extend OEM's monitoring and alerting capabilities to any product in your data center. In this white paper we saw how a custom plug-in can be developed in a couple of simple steps to receive SNMP traps and thereby enabling OEM monitor a product capable of throwing SNMP traps.



Receiving SNMP traps in Oracle Enterprise  
Manager

June 2009

Author: Anirban Chatterjee, Rene Fontcha and  
Dave Kulvete

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
oracle.com



Oracle is committed to developing practices and products that help protect the environment

Copyright © 2009, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

0109