Es ist ganz einfach

Cloud Control Plug-ins selbst gemacht

Ralf Durben
Oracle Deutschland B.V. & Co. KG

Schlüsselworte

Enterprise Manager, Cloud Control,Monitoring, Plug-ins, Erweiterung

Einleitung

Mit Oracle Enterprise Manager Cloud Control lassen sich beliebige Systeme überwachen. Allerdings liefert Oracle die dafür notwendigen Plug-ins nur für die im Oracle Kosmos bekannten Produkte aus. Mittels eines neuen Tools und geschickter Nutzung von Cloud Control Features lassen sich in Kürze Monitoring Plug-ins für beliebige Systeme herstellen.

In diesem Beitrag wird der Weg kurz beschrieben – im Vortrag wird in einer Live-Demo ein Plug-in neu erstellt und in Cloud Control deployed.

Grundprinzip des externen Monitorings in Cloud Control

Als externes Monitoring wird die Verfahrensweise einer Überwachung bezeichnet, bei der von einem System durch Zugriff von extern Daten entnommen werden. Diese Zugriffe können Skripte oder, im Fall einer Datenbank, SQL-Kommandos in einer Datenbanksitzung sein.

Bei Oracle Enterprise Manager Cloud Control wird dieser externe Zugriff vom Enterprise Manager Agenten durchgeführt. Dieser wird technisch gesehen mittels XML-Dateien für das Monitoring konfiguriert. Diese XML-Dateien beinhalten den Namen eines Zieltyps mit allen Metriken samt Zugriffsmethoden, -intervallen und Bewertungsschwellenwerten. Natürlich muss auch auf der OMS Seite für den jeweiligen Zieltyp eine Datenstruktur im Repository geschaffen werden, damit die Monitoringdaten gespeichert werden können.

Ein Monitoring Plug-in beinhaltet also im Wesentlichen folgende Informationen:

- Name des Zieltyps
- Properties zum Anmelden an das Ziel, falls erforderlich
- Metriken
    - Name der Metrik
    - Erfassungsintervall
    - Zugriffsmethode
    - Zugriffskommando
    - Schwellenwerte

Bei den Zugriffsmethoden unterstützt Cloud Control alle gängigen Verfahren, wie Betriebssystemkommandos, SQL, SNMP, HTTP, URLXML, URL Timing, DMS, JDBC, WBEM, JMX, Webservices, REST und WS-Management. Gerade über Betriebssystemkommandos steht der Überwachung eines beliebigen Systems eigentlich nichts im Wege.

Aufbau eines Plug-ins

Ein Plug-in besteht aus mehreren Dateien, die in einer komprimierten Datei mit der Endung „opar“ zusammengefasst sind. Darunter sind die Konfigurationsdateien in einem speziellen XML-Format und die Skriptdateien, mit denen die Metriken abgefragt werden sollen.

Hilfmittel zum Erstellen von Plug-ins

Als Basis Hilfmittel steht das Extensibility Development Kit (EDK) zur Verfügung, welches eine Anleitung, Beispiele und vor allen die Tools zum Erstellen der OPAR-Datei zur Verfügung stellt. Neuerdings wurde diese rein kommandozeilenbasierte Variante durch eine grafische Unterstützung mit dem Namen Plug-in Builder erweitert. Der Plug-in Builder ist ein Modul im JDeveloper und seit Ende August 2013 im EDK enthalten.

Das EDK können Sie im Rahmen des Self Updates innerhalb von Cloud Control herunterladen. Abbildung 1 zeigt die entsprechende Zeile, wobei im vorliegenden System das EDK schon heruntergeladen wurde.

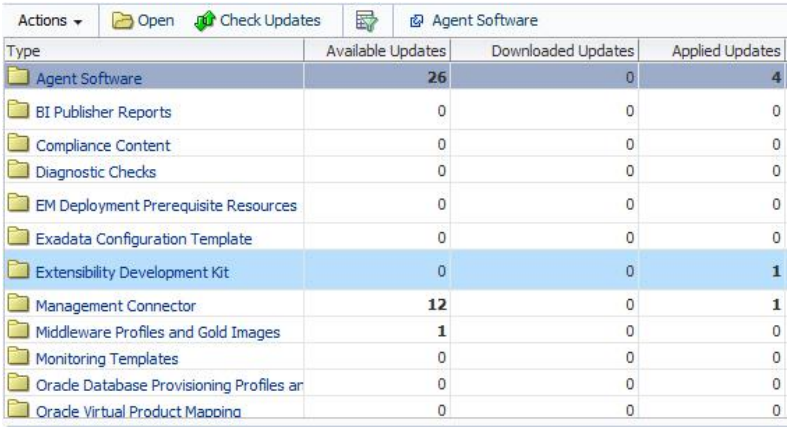

Abb. 1: EDK in Self Update

Nachdem das EDK nun auf der OMS Seite vorliegt, können Sie nun im Menüpunkt „Setup->Extensibility->Development Kit“ weitere Anweisungen aufrufen.

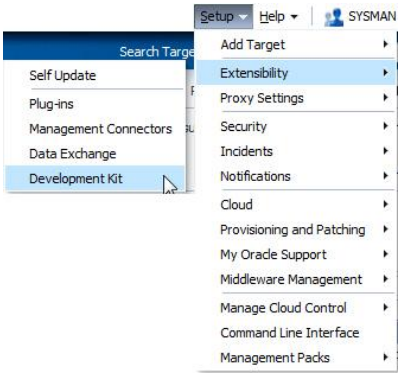

Abb. 2: Auffinden des Development Kits in Cloud Control

Im Bereich “Deployment” finden Sie hinter dem ersten Bullet einen Downloadlink, mit dem Sie das EDK auf Ihren Client Rechner herunterladen können.

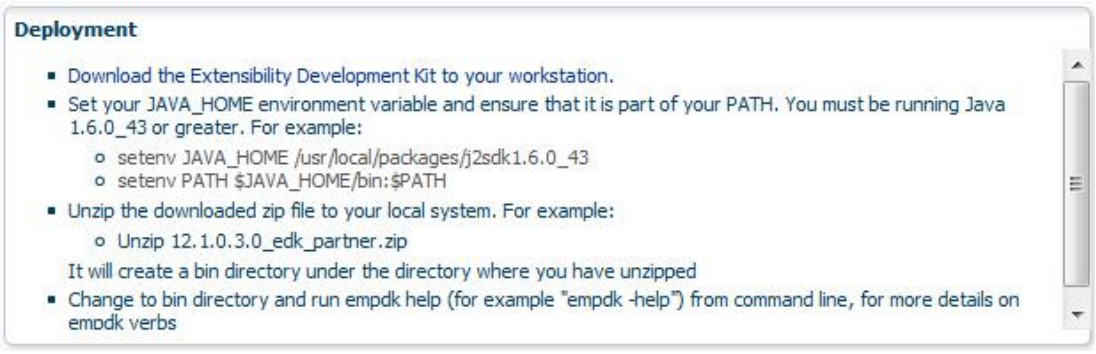

Abb. 3: Der Weg des EDK auf Ihren Cllient Rechner

Jetzt fehlt Ihnen noch das Tool JDeveloper, welches Sie in der Version 11.1.1.7 benötigen. Die Studio Edition wird zwar in den Handbüchern empfohlen, ist aber recht groß und die schlanke Java Edition ist auch ausreichend. Entscheiden Sie sich also für eine dieser Editionen. Der Autor hat im folgenden mit der Java Edition gearbeitet.

Nachdem der JDeveloper installiert ist, entpacken Sie die EDK-Zip Datei, die Sie vom OMS heruntergeladen haben. Sie finden im Unterverzeichnis „doc“/“books“ zwei Handbücher im Format PDF. Im Reference Guide in Kapitel 4 finden Sie die Anweisung, wie Sie den Plug-in Builder in einen bestehenden JDeveloper hineininstallieren. Das sind zum Zeitpunkt der Erstellung dieses Artikels schnelle sieben Schritte.

Den Erfolg können Sie daran messen, dass in JDeveloper nun unter dem Menüpunkt „New“ die Kategorie „Enterprise Manager Plug-in“ auftaucht.

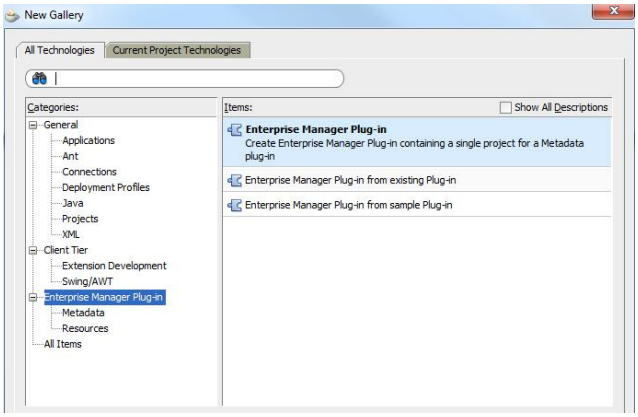

Abb. 4: Die neue Kategorie „Enterprise Manager Plug-in“ in JDeveloper

Erstellen eines eigenen Plug-ins

Wenn Sie im JDeveloper ein neues Projekt der Kategorie „Enterprise Manager Plug-in“ erstellen, werden alle notwendigen Dateien eines Monitoring Plug-ins erzeugt. Sie geben dazu verschiedene Namen (Produktname, Zieltypname, Firmenname,...) an, die später ihre Repräsentierung in den Plug-in Dateien wiederfinden.

Ein Monitoring Plug-in muss mindestens eine Metrik besitzen. Der Name dieser Metrik ist „Response“ und wird zur Lebend-Überwachung verwenden, also um zu sehen, ob ein System verfügbar ist, oder nicht. Im neu erstellten Plug-in ist auch eine solche Metrik enthalten, die in der

Regel aber nicht unverändert verwendbar ist. Hier hilft der Blick in bestehende Plug-ins oder das Handbuch, um eine eigene Metrik zu erstellen.

Die Details dieser Metrik ändern Sie im JDeveloper Projekt unter „Projektname->Resources->oms->metadata->targetType->xxxxx.xml“

Exemplarisch hier ein Beispiel für eine Response Metrik auf Basis eines Perl Skripts:

```
<Metric NAME="Response" TYPE="TABLE">
    <Display>
      <Label NLSID="NLSID_Response">
        Status
      </Label>
    </Display>
    <TableDescriptor>
      <ColumnDescriptor NAME="Status" TYPE="NUMBER" IS_KEY="FALSE">
        <Display>
          <Label NLSID="NLSID_Status">
            Status
          </Label>
        </Display>
      </ColumnDescriptor>
    </TableDescriptor>
    <QueryDescriptor FETCHLET_ID="OSLineToken">
      <Property NAME="perlBin" SCOPE="SYSTEMGLOBAL">perlBin</Property>
      <Property NAME="scriptsDir" SCOPE="SYSTEMGLOBAL">scriptsDir</Property>
      <Property NAME="command" SCOPE="GLOBAL">
        %perlBin%/perl %scriptsDir%/response_mytarget.pl
      </Property>
      <Property NAME="startsWith" SCOPE="GLOBAL">em_result=</Property>
      <Property NAME="warningStartsWith" SCOPE="GLOBAL">em_warning=</Property>
      <Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
    </QueryDescriptor>
  </Metric>
```

Das Perl Skript gibt dazu den entsprechenden Wert mit dem Prefix „em_result=“ aus, also im einfachsten Beispiel sieht das Skript so aus (muss dann unter „Projektname->Resources->agent->scripts“ gespeichert werden):

response_mytarget.pl:

```
print "em_result=3\n"
```

Die Bedeutung dieses Wertes geben Sie als Schwellenwert an unter „Projektname->Resources->oms->default_collection->targetType->xxxxx.xml“. Hier die Anzeige in XML:

```
<CollectionItem NAME="Response">
  <Schedule>
    <IntervalSchedule INTERVAL="1"/>
  </Schedule>
  <Condition COLUMN_NAME="Status" CRITICAL="1" OPERATOR="LT" WARNING="2"/>
</CollectionItem>
```

Die Response Metrik wird also einmal pro Minute erfasst. Ein Wert unter 2 wird als Warnung interpretiert, ein Wert unter 1 gilt als kritisch.

Im Folgenden können nun beliebig viele Metriken erstellt werden – entweder durch die deklarative Angabe im Plugin-Builder, die dieser dann in korrektes XML umwandelt, oder durch direktes Einfügen von XML. Ein weiterer Weg ist aber die Nutzung von Metric Extensions in Cloud Control. Dazu wird das neue Plug-in mit nur der Response Metrik in Cloud Control deployed (siehe unten) und weitere Metriken als „Metric Extension“ erzeugt. Cloud Control bietet dazu eine sehr leicht zu bedienende Oberfläche an. Die neuen Metriken können direkt in Cloud Control getestet werden.

Da für die Metric Extensions letztlich auch nur wieder XML-Dateien angelegt werden, können diese nun herangezogen werden und die Metrikdefinitionen in den Plug-in Builder kopiert werden.

Delopyment von eigenen Plug-ins.

Wenn die einzelnen Definitionen für das Plug-in im Plug-in Builder vorliegen, erzeugen Sie mit einem Rechtsklick auf das Projekt und dem Menüpunkt „Create Plug-in Archive“ eine Datei mit der Endung „opar“.

Diese OPAR-Datei muss mit dem Enterprise Manager Command Line Interface (EMCLI) in das Repository importiert werden. Dazu wird das Kommando emcli import_update verwendet. Ein Beispiel:

```
emcli import_update -file=/pl/12.1.0.1.0_RD.WDG.xwdg_2000_0.opar -omslocal
```

Nach erfolgreichem Upload erscheint das neue Plug-in im Bereich “Setup->Extensibility->Plug-ins” und kann dort in den OMS deployed werden. Interessant ist hierbei, dass ein Deployment der ersten Version ihres neuen Plug-ins auf Agenten nicht notwendig ist. Wenn Sie ein neues Ziel mit ihrem neuen Zieltyp in Cloud Control hinzufügen, wird das neue Plug-in automatisch auf den angegebenen Agenten übertragen.

Aber Achtung: Wenn Sie für ihr eigenes Plug-in eine neue Version erstellen und einspielen, findet hierfür keine automatische Übertragung auf die Agenten statt. In diesem Fall müssen Sie die neue Version manuell mit dem Button „Deploy“ auf die Agenten bringen.

Fazit

Die Erstellung eigener Plug-ins erfordet zwar immer noch einige Handarbeit, ist aber mittlerweile viel einfacher als in früheren Versionen. Nach einer kurzen Einarbeitung können sehr schnell große Erfolge erzielt werden. Damit kann ein Monitoring durch Cloud Control auf viele sonst nicht unterstützte Systeme ausgedehnt werden.

Im Vortrag auf der DOAG Konferenz wird ein neues Plug-in LIVE erstellt.

Kontaktadresse:
Ralf Durben
Oracle Deutschland B.V. & Co. KG
Rudolfstr. 5,
D-59556 Lippstadt

Telefon:	+49 (0) 211-74839 461
E-Mail	ralf.durben@oracle.com
Internet:	www.oracle.com