

DOAG

Deutsche ORACLE -Anwendergruppe e.V.

Konferenz Nürnberg 2014

Den Oracle Enterprise Manager flexibel erweitern

EIN ORACLE NOSQL ENTERPRISE MANAGER / CLOUD CONTROL 12C PLUG-IN ERSTELLEN



NoSQL Vortrag – Taiwan/Taipei 2014

Agenda

- +
- Architektur
- Vorbereitung:
 - Versionsnummer / Namensgebung / Verzeichnisstruktur
- Überwachung auf dem Agent
 - Zu überwachende Metriken für den Agent deklarieren
- Plug-In Homepage erstellen
 - Anzeige der Homepage des Target im OMS
- Weitere Möglichkeiten der Erfassung

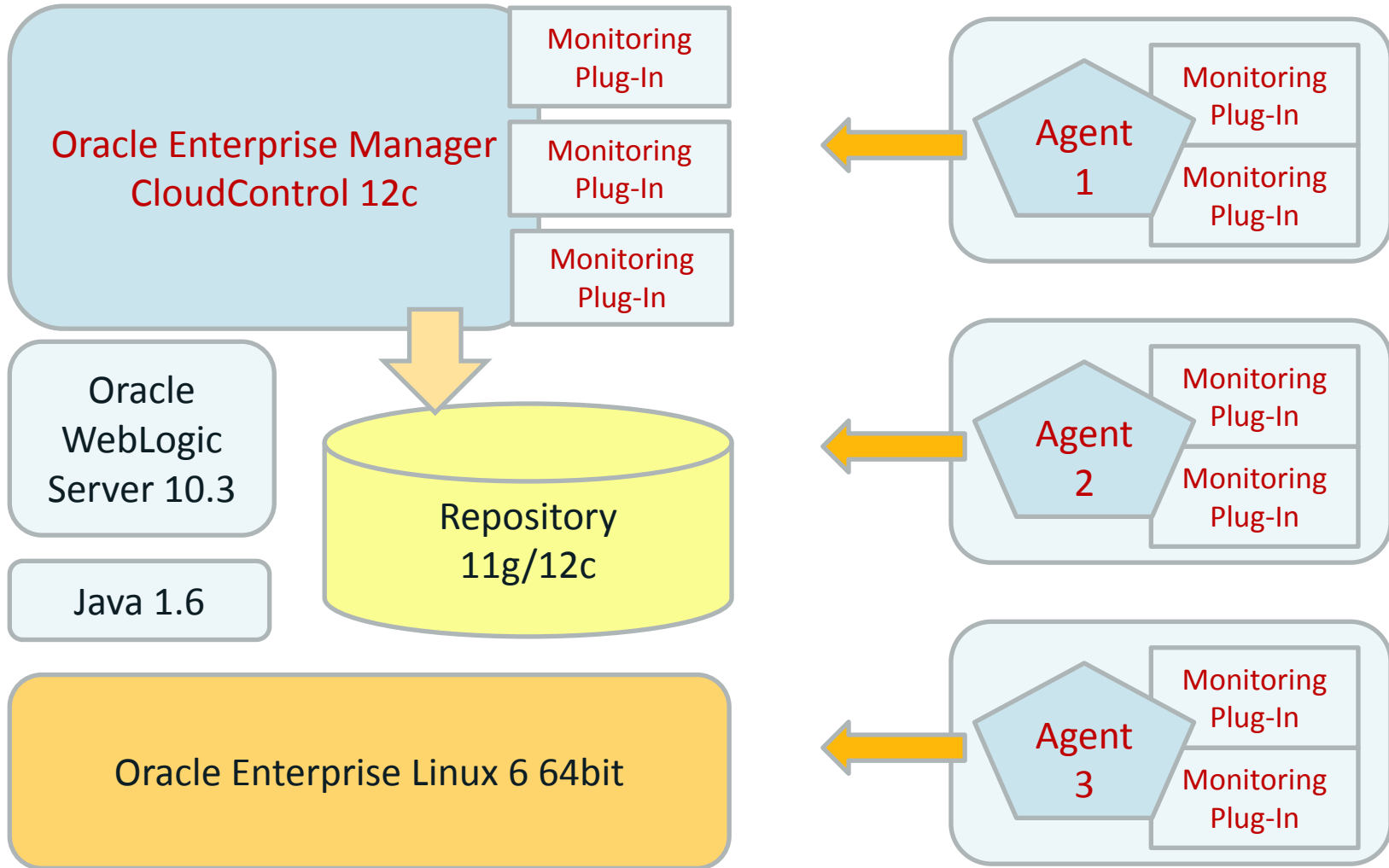




Grundlagen - Konzept

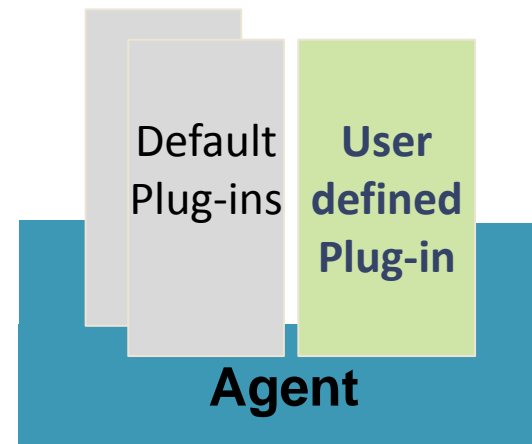
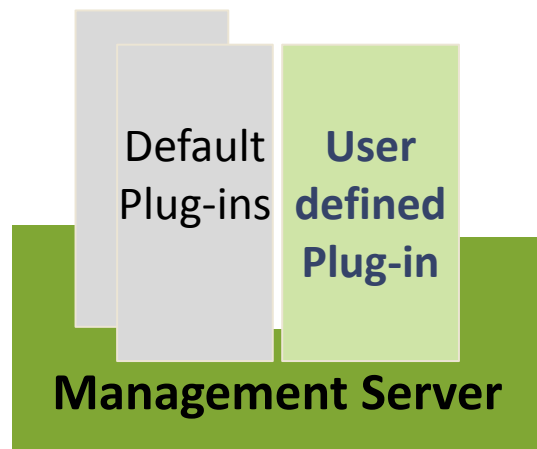
Das Oracle 12c Plug-In Konzept

OMS 12c Architektur - Gesamtübersicht



Architektur Plug-In OMS

- Seit Oracle 12c setzt Cloud Control auf eine konsequente Plug-In Architektur mit der Möglichkeit der deklarativen Programmierung / Entwicklung eigener Erweiterungen



(aktuelle OEM Version 12.1.0.4.0)

Vorteil:

Einzelne Teile (bzw. Targets wie die Datenbank) lassen sich je nach Bedarf getrennt patchen!

Ziel für unser Plug-In (1)

- Für die Oracle NoSQL Datenbank steht noch kein offizielles Plug-In zur Verfügung - Es ist zu erwarten, dass auch nur die kommerzielle Version mit einem Plug-In ausgeliefert wird
- Die Oracle NoSQL Datenbank soll aber nun ebenfalls über CloudControl 12c in der Kundenumgebung überwacht werden
- Zu überwachende Eigenschaften:
 - Ist das System auf dem lokalen Node verfügbar (Status des SN)
 - In welchem Status befinden sich die Nodes/Replica des gesamten Stores
 - Wie viele Datensätze sind im Store abgelegt
 - Wie viel Plattenplatz verbraucht der Store auf der lokalen Maschine und wie viel Plattenplatz steht für den Store noch zur Verfügung
 - Wie hoch ist der Speicherverbrauch der Java Umgebung

Ziel für unser Plug-In (2)

- Eine eigene Home Page im Management Server

[NoSQL_DCGPIDB_on_oraoms12c01.pipperr.local_5000](#) oraoms12c01.pipperr.local
 nosql Overview Page Refreshed Nov 14, 2014 12:49:58 AM CET

Summary

Target Name: NoSQL_DCGPIDB_on_oraoms12
 Target Type: nosql
 Plug Version Nr.: 12.1.0.62.0
 nosql Version Nr.: 12.1.3.0.14
 Current Status: ↑ Up
 Up Since: Nov 14, 2014 12:36 am CET
 Hosted By: oraoms12c01.pipperr.local

Storage Entries

SN or Rep N...	Details of the Node	Statu...	Maste...
sn1	[sn1] dc=dc1 oraoms12c01:5000	RUNNING	
[rg1-rn1] sn=sn1	[rg1]	RUNNING	MASTER
sn2	[sn2] dc=dc1 oraoms12c01:5002	RUNNING	
[rg2-rn1] sn=sn2	[rg2]	RUNNING	MASTER

Store Node Information

Datacenter Name: DCGPIDB
 Store Name: GPIDB
 Store Unique Name: STORE_localhost.localdomain
 Store Unique ID: 6c83c534-6cb9-48d3-ab14-433472be248c
 Root Directory:
 Num Partitions: 20
 Num Cpu Usage: NaN
 Cache Memory: NaN
 HeapMemory UsageMB: 13.603
 Non HeapMemory UsageMB: 12.615
 Total HeapMemory UsageMB: 120.667

Storage Size on this node

166401.191 (Free Size MB inside the store disk (84.8%))
 2.733 (Size MB Used of the disk without the store (15.2%))
 29716.267 (Database Size (MB) on this Node (0%))

Storage Entries

Storage Operations per Minute

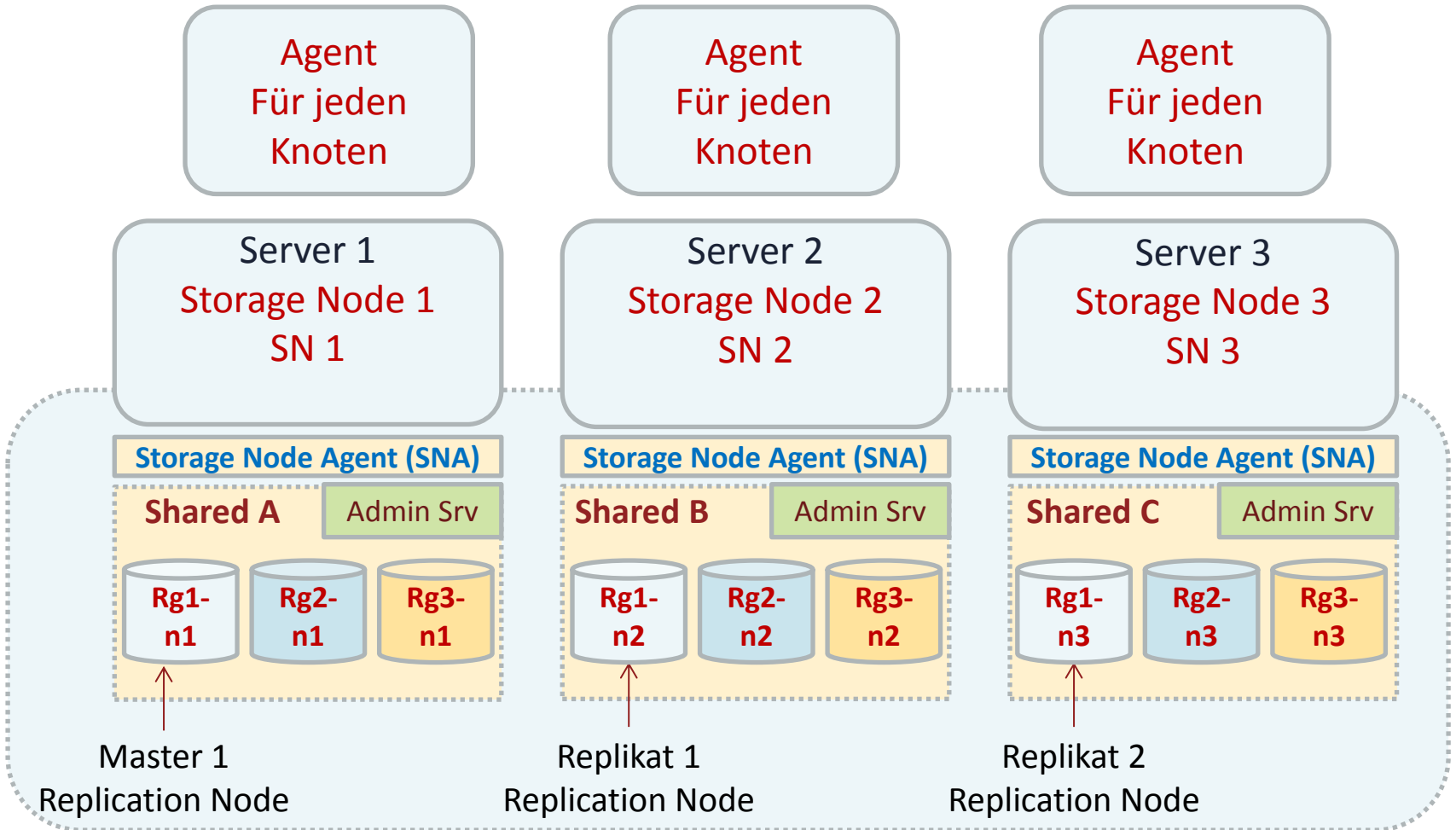
Incidents and Problems

* Target: Local target and related targets | * Category: All | 0 | 0 | 0 | 0

Summary	Target	Severity	Status	Escalation level	Type	Time Since Last Update

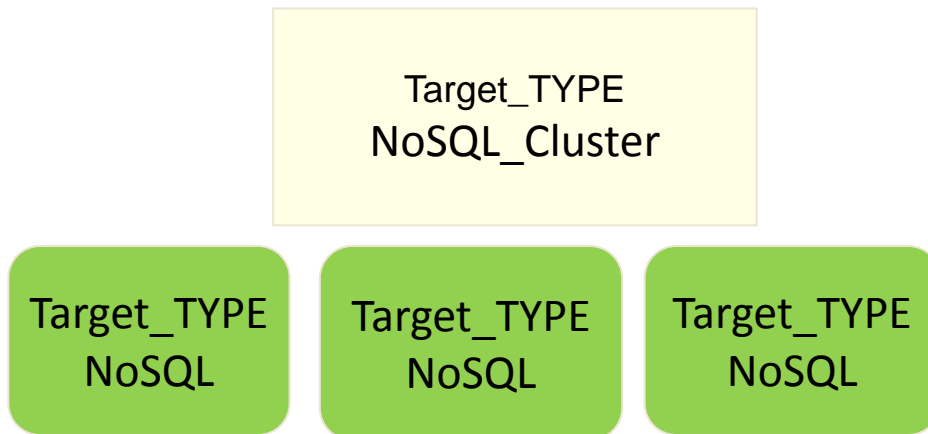
NoSQL Architektur versus Single Server Plug-In

- NoSQL Architektur



NoSQL as a Cluster?

- Im Prinzip stellt ein NoSQL Store in der Überwachung ein Cluster dar
- Ideale Umsetzung:
 - Cluster Homepage für den eigentlichen Store
 - Metadaten des Stores wie Anzahl der Einträge
 - Eine Seite pro Node



Ein Problem:

- Unique ID im NoSQL Store fehlt per Default

Hauptproblem:

- Erstellen von Cluster Plug-Ins nicht dokumentiert
- Ableiten aus dem DB Plug-In nicht erfolgreich

Workaround über Redundancy System

- ✦ ■ Plug-In für den Target Type "NoSQL" ✦
 - Jeder Knoten wird als eigenständiges Target erfasst
 - Nachteil: Anzahl der Einträge wird doppelt ermittelt
 - Beim Erfassen einer bestimmten Metrik auf dem Knoten, wird im Store eine unique Identifier angelegt, falls diese noch nicht existiert
 - Damit lassen sich später die Knoten wieder zu ihren Stores zusammenfassen
 - Per Hand wird ein " Redundancy System" angelegt und die passenden Knoten werden diesem System zugeordnet

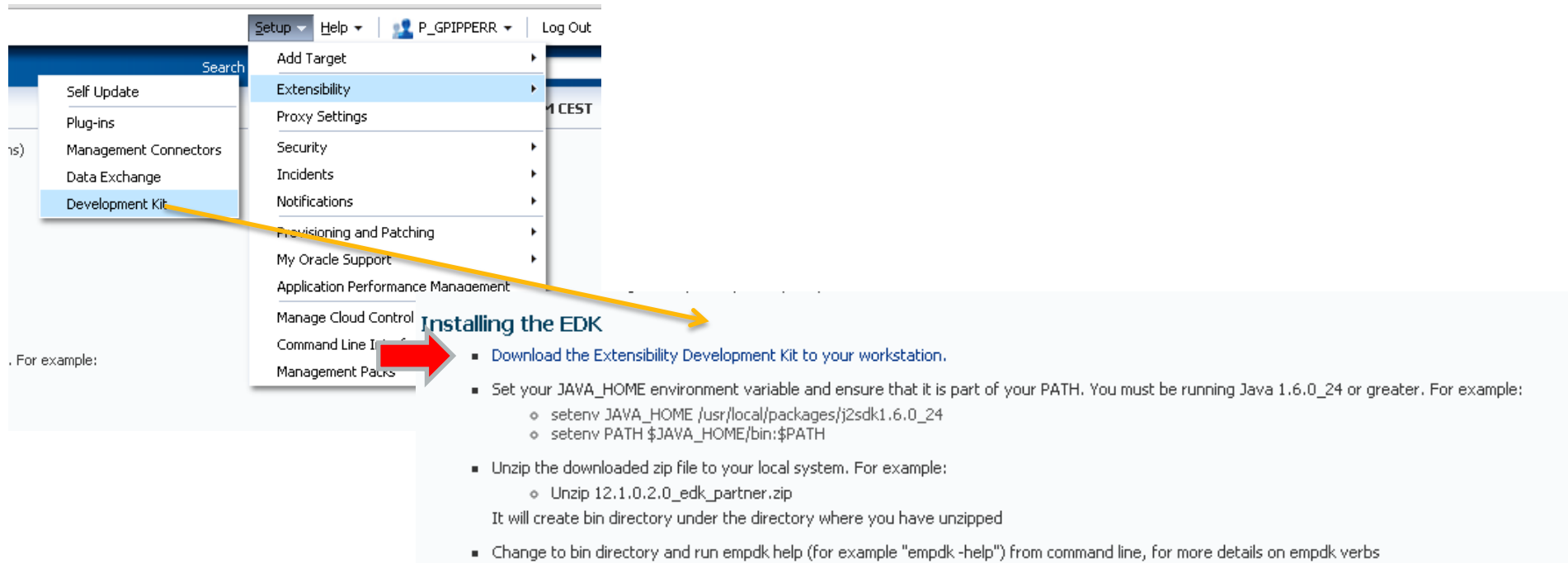


Aufsetzen der Entwicklungsumgebung

Ordnung muss sein

Aufsetzen der Entwicklungs-Umgebung

- Das „Extensibility Development Kit“ - EDK - für die Plug-In Entwicklung vom OMS laden



Installing the EDK

- 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 1.6.0_24 or greater. For example:
 - setenv JAVA_HOME /usr/local/packages/j2sdk1.6.0_24
 - setenv PATH \$JAVA_HOME/bin:\$PATH
- Unzip the downloaded zip file to your local system. For example:
 - Unzip 12.1.0.2.0_edk_partner.zipIt will create 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

– Voraussetzung:

- Java **1.6**
- Unter Windows – Lokale Admin Rechte!

Um das EDK mit EMCLI zu laden:
emcli get_ext_dev_kit

Aufsetzen der Entwicklungs-Umgebung

- Das Extensibility Development Kit“ - EDK - enthält alle notwendigen Dateien, um ein Plug-In aus den Definitions Dateien zu erzeugen
- Zip File des EDK in ein lokales Verzeichnis "edk" extrahieren
 - Zum Beispiel unter ~/plugin_dev/edk

```
unzip 12.1.0.4.0_edk_partner.zip -d ~/plugin_dev/edk/
```

Die Plug-In Entwicklung erfolgt am bestem auf dem Management Server selbst.

Vorbereitung – Verzeichnis Struktur Projekt

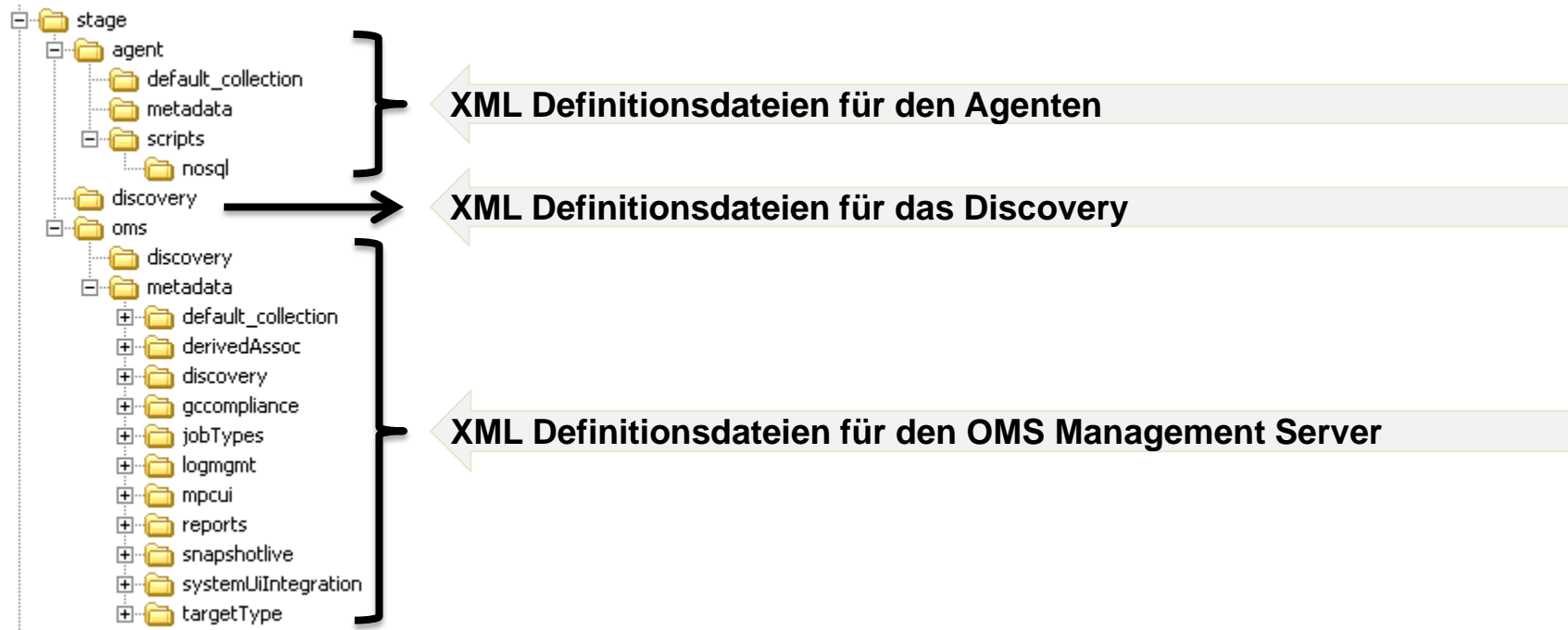
- Die Projektstruktur kann frei gewählt werden – für unser Projekt verwenden wir:



Struktur und Namensgebungen immer sehr genau kontrollieren!

Vorbereitung – Struktur für das Plug-In anlegen

- Die deklarative Entwicklung über XML Definitionsdateien erfordert eine streng einzuhaltende Verzeichnisstruktur



Zu Beginn die häufigste Fehlerquelle!

Vorbereitung – Versions-Nr. zweistellig definieren

- Wichtig ist die Einhaltung der Syntax bei der Versionsnummer

Plug-In Version definieren: **a.b.c.d.e**

a.b = Die Version des Enterprise Manager Extensibility Development Kit (EDK),
dass bei der Entwicklung verwendet wurde (**12.1**, 12.2, usw.).

c = Muss immer eine **0** sein

d = Die aktuelle Version des Plug-In, vom Entwickler vergeben
+ können 1 oder 2 Digits ($[1-9] \mid [0-9][1-9]$) sein
+ Bei jedem Update muss die Nummer angepasst werden

e = Für den zukünftigen Einsatz - Default Wert ist **0**

Die Plug-In Version lautet damit für die erste Version: **12.1.0.01.0**

Wichtig ist es, dass die letzte Stelle NICHT verwendet wird!

Ist die Versionsnummer falsch (zum Beispiel weil die letzte Nr. eine 0 ist) schlägt das Deployment fehl!

Vorbereitung – Plug-In ID und Target Type

- Die Plug-In **ID identifiziert eindeutig** das selbst-entwickelte Plug-In
- Plug-In ID definieren
 - Vendor ID (8 chars) = gpiconsl
 - Product ID (8 chars) = nosql
 - Plug-in Tag (4 chars) = **x001** **Muss ab 12.1.0.4 zwingend ein x sein!**
- Der Plug-In Name lautet damit **gpiconsl.nosql.x001**
- Kürzel für den „Target Type“ vergeben: **nosql**

Streng auf die durchgängige Verwendung und richtige Benennung des „Target Type“ achten!

Die XML Definitionen für die ersten Schritte erstellen

■ Plug-In Basisstruktur – Datei und Verzeichnisse

– plugin.xml

- AGENT

- plugin_registry.xml
- default_collection/nosql.xml
- metadata/nosql.xml

- DISCOVERY

- OMS

- metadata/targetType/nosql.xml
- metadata/default_collection/nosql.xml

Für eine Basisversion sind 4 XML Dateien ausreichend

Der Name nosql entspricht dem Target Namen
– empfohlene Namensgebung!

1 zu 1 Kopie aus dem Agent Home!

Im ersten Test wurde ein Fehler geworfen, wenn diese Dateinamen NICHT gleich zum Target Type Namen sind!

Umsetzung – **Plugin.xml** – Definitionsdatei für das Plug-In

- Speicherort: <stage>/**plugin.xml**
- Zu definierende Werte:
 - Plug-In ID
 - Plug-In Version
 - Readme und Description
 - Target Type Name und welche Versionen des eigenen Plug-Ins werden unterstützt
 - Auf welcher OEM Version kann das Plug-In installiert werden
 - Plug-In Attribute
 - Auf welchen Plattformen läuft das Plug-In

Umsetzung – Beispiel für die `plugin.xml` (1)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Plugin xmlns="http://www.oracle.com/EnterpriseGridControl/plugin_metadata">
  <PluginId pluginTag="x001" productId="nosql" vendorId="opicons1"/>
  <PluginVersion value="12.1.0.10.0"/>
  <ShortDescription>Oracle nosql Plugin for monitoring Oracle nosql Store from the Enterprise Manager</ShortDescription>
  <Readme><![CDATA[
    The nosql plugin supports monitoring of Oracle nosql Storage Nodes.
    This covers monitoring of the nosql Node availability and performance and various J2EE specific metrics.
    Target Typ nosql for each Host
  ]]></Readme>
  <TargetTypeList>
    <TargetType isIncluded="TRUE" name="nosql">
      <VersionSupport>
        <SupportedVersion supportLevel="Comprehensive" maxVersion="2.0.0.0" minVersion="1.0.0.0"/>
        <!--NonSupportedVersion minVersion="1.0.0.0"/ -->
      </VersionSupport>
    </TargetType>
  </TargetTypeList>
  <EMPlatforms>
    <CertifiedPlatform version="12.1.0.4.0"/>
  </EMPlatforms>
  <PluginAttributes DisplayName="Oracle nosql"
    Category="Databases"
    Type="MP"
    ReleaseDate="2014-08-26"
    ReleaseStatus="Beta"
    OnlineDocLink="http://www.pipperr.de/dokuwiki/doku.php?id=dba:cloudcontrol_12c_oracle_install"
    LastUpdDate="2014-08-26"
  />
```

Plug-In ID

Plug-In Version

Target Type

Plug-In wird als Database Plug-In definiert

Umsetzung – Beispiel für die `plugin.xml` (2)

```
<PluginOMSOSAruId value="2000"></PluginOMSOSAruId>
```

```
<Certification>
```

```
  <Component type="OMS">
```

```
    <CertifiedPorts>
```

```
      <PortARUID value="46" />
```

```
      <PortARUID value="226" />
```

```
    </CertifiedPorts>
```

```
  </Component>
```

```
  <Component type="Agent">
```

```
    <CertifiedPorts>
```

```
      <PortARUID value="46" />
```

```
      <PortARUID value="226" />
```

```
    </CertifiedPorts>
```

```
  </Component>
```

```
  <Component type="Discovery">
```

```
    <CertifiedPorts>
```

```
      <PortARUID value="46" />
```

```
      <PortARUID value="226" />
```

```
    </CertifiedPorts>
```

```
  </Component>
```

```
</Certification>
```

```
</Plugin>
```

OS der Oracle Management Servers – 2000 Generisch

OS der Agent Betriebssysteme - 46 / 226 für Linux 32 /Linux 64

Umsetzung – agent/plugin_registry.xml

- Mit der Datei `plugin_registry.xml` im Agent Verzeichnis wird das eigentliche Agent Plug-In definiert
 - Werte
 - PlugID - muss der ID aus `plugin.xml` entsprechen
 - Speicherort für die Metrik Definitionsdateien

```
<?xml version="1.0"?>
<PlugIn ID="gpiconsl.nosql.x001" Version="12.1.0.10.0" HotPluggable="false" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.oracle.com/EnterpriseGridControl/plugin plugin.xsd">
  <TargetTypes>
    <FileLocation>metadata/nosql.xml</FileLocation>
  </TargetTypes>

  <TargetCollections>
    <FileLocation>default_collection/nosql.xml</FileLocation>
  </TargetCollections>

  <TypeDefaults>
    <Type Name="nosql" Prefix="false" MetricCacheUsage="Medium" />
  </TypeDefaults>

</PlugIn>
```

AGENT Umsetzung – agent/metata/nosql.xml

- In der Datei `nosql.xml` werden die Metriken definiert
- Wichtiger Parameter `META_VER`
 - Bei der Entwicklung kann die Daten manuell auf dem OMS deployed werden
 - Damit aber eine Veränderung vom OMS erkannt werden kann, muss die Versions-Nummer dieser Datei bei jedem Upload hoch gezählt werden

Bei 0.1 beginnen und bei jeder Aktualisierung hochzählen

```
<TargetMetadata META_VER="0.1" TYPE="nosql" CATEGORY_PROPERTIES="VersionCategory">
```

Auf die korrekte Schreibweise des Target Type achten!

Metriken (Checks) für den Agenten definieren

- Mit den **zwei XML Dateien** im Verzeichnis "**metadata**" und "**default_collection**" werden die Metriken bzw. Checks definiert

- In der **nosql.xml** im Verzeichnis "**metadata**" :

WIE und **WELCHE** Werte sollen ermittelt werden

- In der **nosql.xml** im Verzeichnis "**default_collection**":

WAS und **WIE OFT** soll zum OMS versandt werden

Metrik Nr. 1 – Response (Ist der Dienst verfügbar) (1)

- In der metric XML (in unseren Fall die "metadata/nosql.xml") im "metadata" Verzeichnis wird definiert, **WIE** ein Wert auf dem System ermittelt werden soll
- Dazu werden die Oracle "**Fetchlets**" verwendet
 - **Fetchlets** => Anweisungsblöcke mit welcher Technologie und wie die Werte ermittelt werden können (13 Typen stehen zur Verfügung)
 - Eingesetzte Fetchlet Typen im diesen Plug-In
 - JMX Abfragen (JAVA Management Interface abfragen)
 - **FETCHLET_ID="JMX"**
 - Script Aufrufe (mit Hilfe von frei definierbaren Bash/perl Skripten werden Daten erfasst und als Text-Zeilen zurückgegeben)
 - **FETCHLET_ID="OSLineToken"**

Metrik Nr. 1 – Response (Ist der Dienst verfügbar) (2)

```
<Metric NAME="Response" TYPE="TABLE">
  <Display>
    <Label NLSID="Response">Response</Label>
  </Display>
  <TableDescriptor>
    <ColumnDescriptor NAME="StartTime" TYPE="NUMBER" IS_KEY="FALSE" TRANSIENT="TRUE">
      <Display>
        <Label NLSID="StartTime">Start Time</Label>
        <Description NLSID="StartTimeDescription">Start time of the NoSQL server</Description>
      </Display>
    </ColumnDescriptor>
    <ColumnDescriptor NAME="Status" TYPE="NUMBER" COMPUTE_EXPR="(StartTime>0)?StartTime/StartTime:0" COLUMN_NAME="Status">
      <Display>
        <Label NLSID="Status">Status</Label>
        <Description NLSID="StatusDescription">Status of the Oracle NoSQL Node - whether up or down</Description>
      </Display>
    </ColumnDescriptor>
  </TableDescriptor>
  <QueryDescriptor FETCHLET_ID="JMX">
    <Property NAME="MachineName" SCOPE="INSTANCE">Host</Property>
    <Property NAME="Port" SCOPE="INSTANCE">JMXPort</Property>
    <Property NAME="UserName" SCOPE="INSTANCE" OPTIONAL="TRUE">JMXUserName</Property>
    <Property NAME="password" SCOPE="INSTANCE" OPTIONAL="TRUE">JMXPassword</Property>
    <Property NAME="protocol" SCOPE="INSTANCE">JMXProtocol</Property>
    <Property NAME="service" SCOPE="INSTANCE">JMXService</Property>
    <Property NAME="SSLTrustStore" SCOPE="INSTANCE" OPTIONAL="TRUE">SSLTrustStore</Property>
    <Property NAME="SSLTrustStorePassword" SCOPE="INSTANCE" OPTIONAL="TRUE">SSLTrustStorePassword</Property>
    <Property NAME="valueWhenDown" SCOPE="GLOBAL">0</Property>
    <Property NAME="metric" SCOPE="GLOBAL">java.lang:type=Runtime</Property>
    <Property NAME="columnOrder" SCOPE="GLOBAL">StartTime</Property>
  </QueryDescriptor>
</Metric>
```

True bedeutet, Wert wird NICHT in der Metrik View angezeigt

Spalten Definition

Der Status wird aus der StartTime Spalte errechnet

JMX Parameter

metadata/nosql.xml

Metrik Nr. 1 – Response (Ist der Dienst verfügbar) (3)

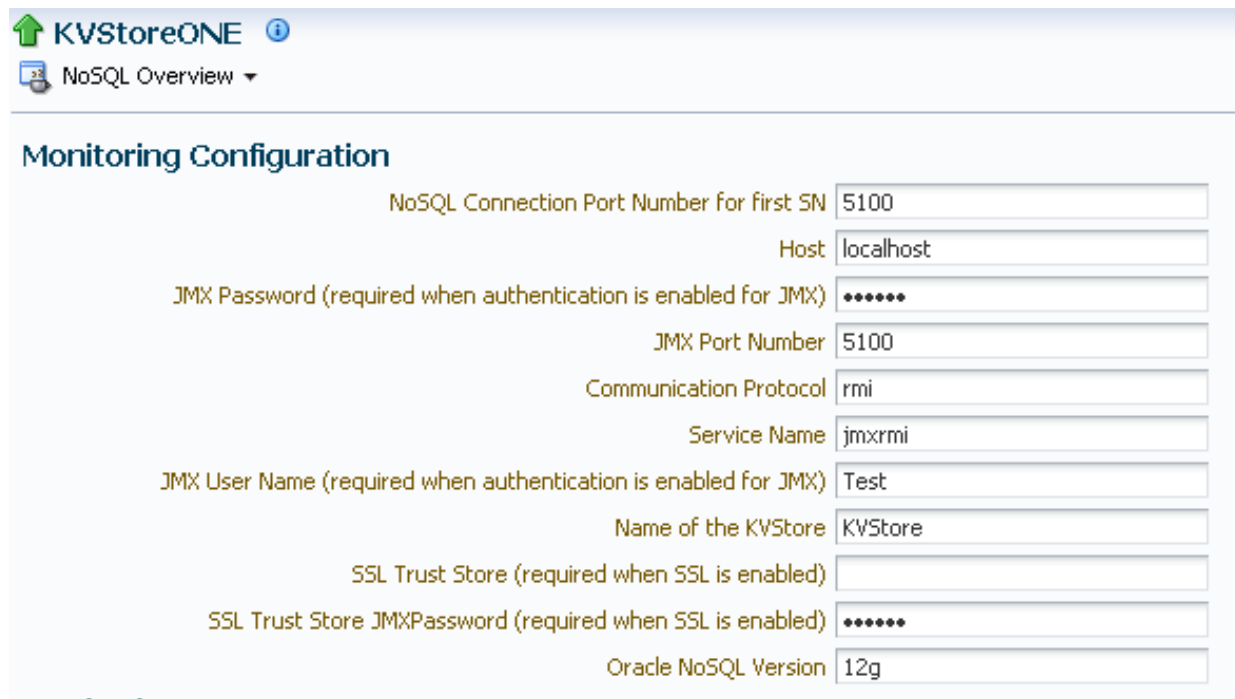
- Wie kommen wir zu den Parameter für das Fetchlet?
- Parameter für Setup und Credential werden ebenfalls in der "[metadata/nosql.xml](#)" Datei definiert

```
<InstanceProperties>
  <InstanceProperty NAME="Host">
    <Display>
      <Label NLSID="NoSQL_HOST_LABEL">Host</Label>
    </Display>
  </InstanceProperty>
  <InstanceProperty NAME="DBPort">
    <Display>
      <Label NLSID="DB_Port_LABEL ">NoSQL Connection Port Number for first SN</Label>
    </Display>
  </InstanceProperty>
  <InstanceProperty NAME="KVStoreName">
    <Display>
      <Label NLSID="NoSQL_KVS_LABEL">Name of the KVStore</Label>
    </Display>
  </InstanceProperty>
  <InstanceProperty NAME="JMXPort">
    <Display>
      <Label NLSID="jmx_Port_LABEL ">JMX Port Number</Label>
    </Display>
  </InstanceProperty>
</InstanceProperties>
```

metadata/nosql.xml

Metrik Nr. 1 – Response (Ist der Dienst verfügbar) (4)

- Aus diesen Parametern entsteht auch die Maske im OMS bzgl. der Monitoring Konfiguration:



KVStoreONE ?

NoSQL Overview ▾

Monitoring Configuration

NoSQL Connection Port Number for first SN	<input type="text" value="5100"/>
Host	<input type="text" value="localhost"/>
JMX Password (required when authentication is enabled for JMX)	<input type="password" value="*****"/>
JMX Port Number	<input type="text" value="5100"/>
Communication Protocol	<input type="text" value="rmi"/>
Service Name	<input type="text" value="jmxrmi"/>
JMX User Name (required when authentication is enabled for JMX)	<input type="text" value="Test"/>
Name of the KVStore	<input type="text" value="KVStore"/>
SSL Trust Store (required when SSL is enabled)	<input type="text"/>
SSL Trust Store JMXPassword (required when SSL is enabled)	<input type="password" value="*****"/>
Oracle NoSQL Version	<input type="text" value="12g"/>

Umsetzung – `nosql.xml` im Verzeichnis `default_colletion`

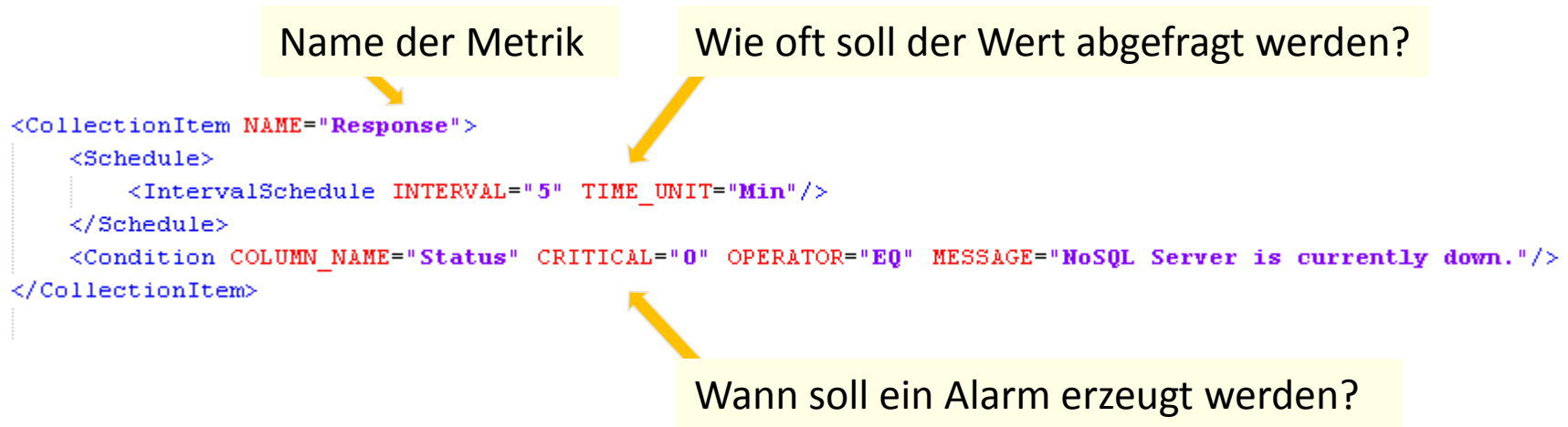
- Die Target Collection definiert **WAS** und **WIE** oft die Daten abgeholt werden
- Definiert die Thresholds für die Alarmierung

```
<TargetCollection TYPE="nosql" INCLUDE_DEFAULT="TRUE">
```

Auf die korrekte Schreibweise der Target Type achten!

AGENT Umsetzung – Metrik bekannt geben

- Über die `nosql.xml` unter "`default_colletion`" wird definiert, **WAS** und **WIE OFT** versandt wird



OSLineToken Fetchlet verwenden

- Über das **OSLineToken Fetchlet** können aus dem Betriebssystem Werte gelesen werden
- Als Rückgabewert wird eine Art CSV Liste erwartet

```
<QueryDescriptor FETCHLET_ID="OSLineToken">
```

```
<Property NAME="scriptsDir" SCOPE="SYSTEMGLOBAL">scriptsDir</Property>
```

```
<Property NAME="command" SCOPE="GLOBAL">/bin/sh</Property>
```

```
<Property NAME="Port" SCOPE="INSTANCE">DBPort</Property>
```

```
<Property NAME="script" SCOPE="GLOBAL">%scriptsDir%/nosql/getKVRoot.sh %Port%</Property>
```

```
<Property NAME="startsWith" SCOPE="GLOBAL">StoreSize=</Property>
```

```
<Property NAME="delimiter" SCOPE="GLOBAL">|</Property>
```

```
</QueryDescriptor>
```

Wert aus Instance Konfiguration verwenden

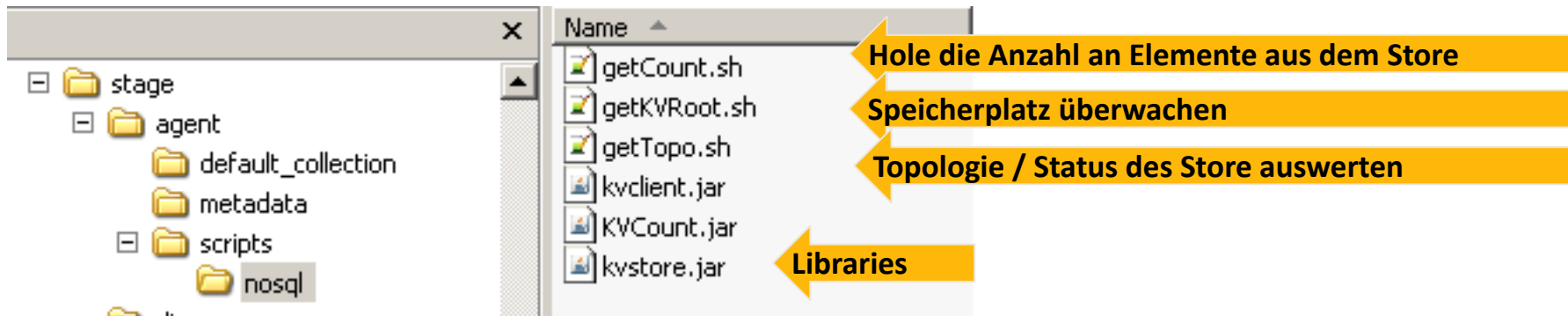
NoSQL Connection Port Number for first SN

Skript erzeugt den folgenden OutPut: **StoreSize=**/u00/app/oracle/kvdata/NODE01/KVStore|21484663|64092696|26035680

metadata/nosql.xml

OSLineToken Fetchlet – Skripte einbinden

- Die aufzurufenden Skripte und alle notwendigen Libraries (wie JAR Files) dazu werden unter agent/scripts/nosql abgelegt



metadata/nosql.xml

OSLineToken Fetchlet – Beispiel Skript

■ Skript für die Größe des Stores:

```
#!/bin/sh
#
# Purpose
# get the kv root
#
##### Environment #####
SCRIPTPATH=$(cd ${0%/*} && echo $PWD/${0##*/})
SCRIPTS_DIR=$(dirname "$SCRIPTPATH")

if test -z "$1"
then
    export KVROOT="NOT_SET"
else
    export KVROOT=$1
fi

echo "use for the Script getKVRoot.sh ${KVROOT} Directory:: ${SCRIPTS_DIR}" >/tmp/log_nosql_agent_script_getkvRoot

if [[ "${KVROOT}" == "NOT_SET" ]]; then
    KVROOTDIR=$(ps afx | grep "kvstore.jar start" | grep "root" | awk '{print $11}' | head -1)
else
    KVROOTDIR="${KVROOT}"
fi

KVROOTDIRSIZE=$(du -sb ${KVROOTDIR} | awk '{ print $1}')
```

Absoluten Pfad ermitteln

```
DISKSIZEUSED=$(df -P ${KVROOTDIR} | grep "/" | awk '{print $3}')
```

```
DISKSIZEFREE=$(df -P ${KVROOTDIR} | grep "/" | awk '{print $4}')
```

```
#result
echo "StoreSize=${KVROOTDIR}|${KVROOTDIRSIZE}|${DISKSIZEUSED}|${DISKSIZEFREE}"
```

Script erzeugt den folgenden OutPut: **StoreSize=/u00/app/oracle/kvdata/NODE01/KVStore|21484663|64092696|26035680**
agent/scripts/nosql/getKVRoot.sh

Agent Metrik - Zusammenfassung

- + ■ Zusammenfassung:
Welche Schritte sind für eine neue Metrik notwendig:
 - **Definition** der Metrik in der "[metadata/nosql.xml](#)"
 - **Bekanntgabe** der Metrik in der "[default_collection/nosql.xml](#)"
 - Kontrolle, ob noch weitere Credentials als **Parameter** in der "[metadata/nosql.xml](#)" hinterlegt werden müssen
 - Kontrolle, ob noch weitere **Parameter als Instance Parameter** in der "[metadata/nosql.xml](#)" hinterlegt werden müssen



Die Version 1 deployen

Wie bringe ich das nun in den OMS?

OEM prüfen

- Vor dem Deploy prüfen, ob der OEM für den Upload konfiguriert worden ist

ORACLE Enterprise Manager Cloud Control

Software Library

Upload File Locations | Referenced File Locations

Configure the storage locations that can be used for uploading files for Software Library entities.

Storage Type: OMS Shared File System

Configure file system locations on the OMS hosts. Make sure the locations are accessible by all the OMS instances, typically mounted or shared with a common credential to be used by the Software Library for reading/writing from/to a location.

Name	Status	Location	Associated Entities	Total Space	Available Space
default_loc	Active	/opt/oracle/swlib/	Show	191.5...	16

Actions: Create Entity, Create Folder, Delete Folder..., Change Maturity, Create Like..., Grant/Revoke Privileges, Move Entity, Add Notes, Add Attachments, Stage Entity, Import, Export, Administration

Version 1 deployen

- Nach dem Erstellen der ersten XML Dateien können wir die erste Version auf dem Management Server ausrollen
- Ablauf:
 - Validieren ob alle XML den Oracle Standards entsprechen
 - `${EMDK_HOME}/bin/empdk validate_plugin ...`
 - OPAR Datei `12.1.0.01.0_gpiconslnosql.x001_2000_0.opar` für das Deployment erzeugen:
 - `${EMDK_HOME}/bin/empdk create_plugin ...`
 - OPAR Datei deployen:
 - `${OEM_HOME}/bin/emcli import_update ...`

Für diese Schritte empfiehlt es sich entsprechende Skripte zu erstellen!

Plug-In auf dem OMS und dem Agent "ausrollen,, - 1

- Plug-In auf dem OMS deployen

plug-ins available, downloaded, and deployed

Plug-ins

This page lists the plug-ins available, downloaded, and actions. Use this page to deploy or undeploy plug-ins.

Name	Version			Management Agent with Plug-in	Description
	Latest Available	Latest Downloaded	On Management Server		
Applications					
Oracle Fusion Applications	12.1.0.6.0	12.1.0.6.0		0	Enterprise Manager for Fusion App
Oracle Siebel	12.1.0.5.0	12.1.0.5.0		0	Enterprise Manager for Oracle Sieb
Cloud					
Databases					
IBM DB2 Database	12.1.0.2.0	12.1.0.2.0		0	IBM DB2 Database Plugin for monit
Microsoft SQLServer Database	12.1.0.4.0	12.1.0.4.0		0	Enterprise Manager for Microsoft S
Oracle Database	12.1.0.6.0	12.1.0.6.0	12.1.0.6.0	1	Enterprise Manager for Oracle Data etc.
Oracle NoSQL Store	12.1.0.01.0	12.1.0.01.0		0	Oracle NoSQL Plugin for monitoring
Sybase ASE Database	12.1.0.2.0	12.1.0.2.0		0	Sybase ASE Server Plugin for monit

Oracle NoSQL Store

General | Recent Deployment Activities

Plug-in ID GPICONSLS.nosql.xmon
Vendor GPICONSLS
Version on Management Server None
Latest Available Version 12.1.0.01.0

Versions Downloaded 12.1.0.01.0
Description Oracle NoSQL Plugin for monitoring Oracle NoSQL Store from the Enterprise Manager

Plug-In auf dem OMS und dem Agent "ausrollen,, - 2

■ Plug-In auf dem OMS deployen

Progress bar: Plug-ins | Prerequisite Checks | Repository | Review | Confirmation

Deploy Plug-ins on Management Servers: Plug-ins

+ Add... - Remove

Name	Latest Available	Currently Deployed	Version	To Deploy
Oracle NoSQL Store	12.1.0.01.0	None		12.1.0.01.0

Oracle NoSQL Store

Target Types	
Name	Supported Target Versions
NoSQL	Plug-in Version - 12.1.0.01.0 1.0.0.0 - 2.0.0.0

What's New in 12.1.0.01.0

Oracle NoSQL Plugin for monitoring Oracle NoSQL Store from the Enterprise Manager

Readme

The NoSQL plugin supports monitoring of Oracle NoSQL Storage Nodes. This covers monitoring of the NoSQL Node availability and performance metrics.

Target Type: NoSQL for each Host

Deploy Plug-ins on Management Servers: Prerequisite Checks

⚠ Prerequisite checks are currently running in a background process. Clicking Cancel or closing browser window will not stop the prerequisite process.

Prerequisite Status

🕒 Prerequisite Checks - In Progress

Name	Version	Status (%)
Oracle NoSQL Store	12.1.0.01.0	25

Plug-In auf dem OMS und dem Agent "ausrollen,, - 3

■ Plug-In auf dem OMS deployen

The screenshot shows the Oracle OMS deployment interface. At the top, a progress bar indicates the current step is 'Prerequisite Checks', with other steps being 'Plug-ins', 'Repository', 'Review', and 'Confirmation'. Below the progress bar, the title is 'Deploy Plug-ins on Management Servers: Prerequisite Checks'. A section titled 'Prerequisite Status' shows a green checkmark and the text 'Prerequisite Checks - Successfully Completed'. A table lists the plug-in 'Oracle NoSQL Store' with version '12.1.0.01.0'. Below this, a section titled 'Deployment Steps: Oracle NoSQL Store' lists various steps with their status and log file locations. A red arrow points from the 'Oracle NoSQL Store' row in the table down to the 'Validate plug-in home' step. Below the deployment steps, the interface transitions to 'Deploy Plug-ins on Management Servers: Repository', which includes a 'SYS Credential' section for specifying repository SYS credentials. This section has radio buttons for 'Named' and 'New' (selected), and input fields for 'Username' (SYS), 'Password', and 'Confirm Password' (all masked with dots). A 'Role' dropdown is set to 'SYSDBA', and a 'Save As' field contains 'NC_SYS'. A red arrow points from the right side of the 'Save As' field towards the right edge of the page.

Name	Version	Status
Oracle NoSQL Store	12.1.0.01.0	

Prerequisite Name	Management Servers	Status
Submit job for running prerequisites check	oraoms12c01.pipperr.local:4889_Manager	✓ Log File
Initialize		✓ Log File
Install software		✓ Log File
Validate plug-in home		✓ Log File
Perform custom preconfiguration		
Check mandatory patches		
Generate metadata SQL		
Preconfigure Management Repository		

Deploy Plug-ins on Management Servers: Repository

▲ SYS Credential

Specify repository SYS credentials

Credential Named New

* Username

* Password

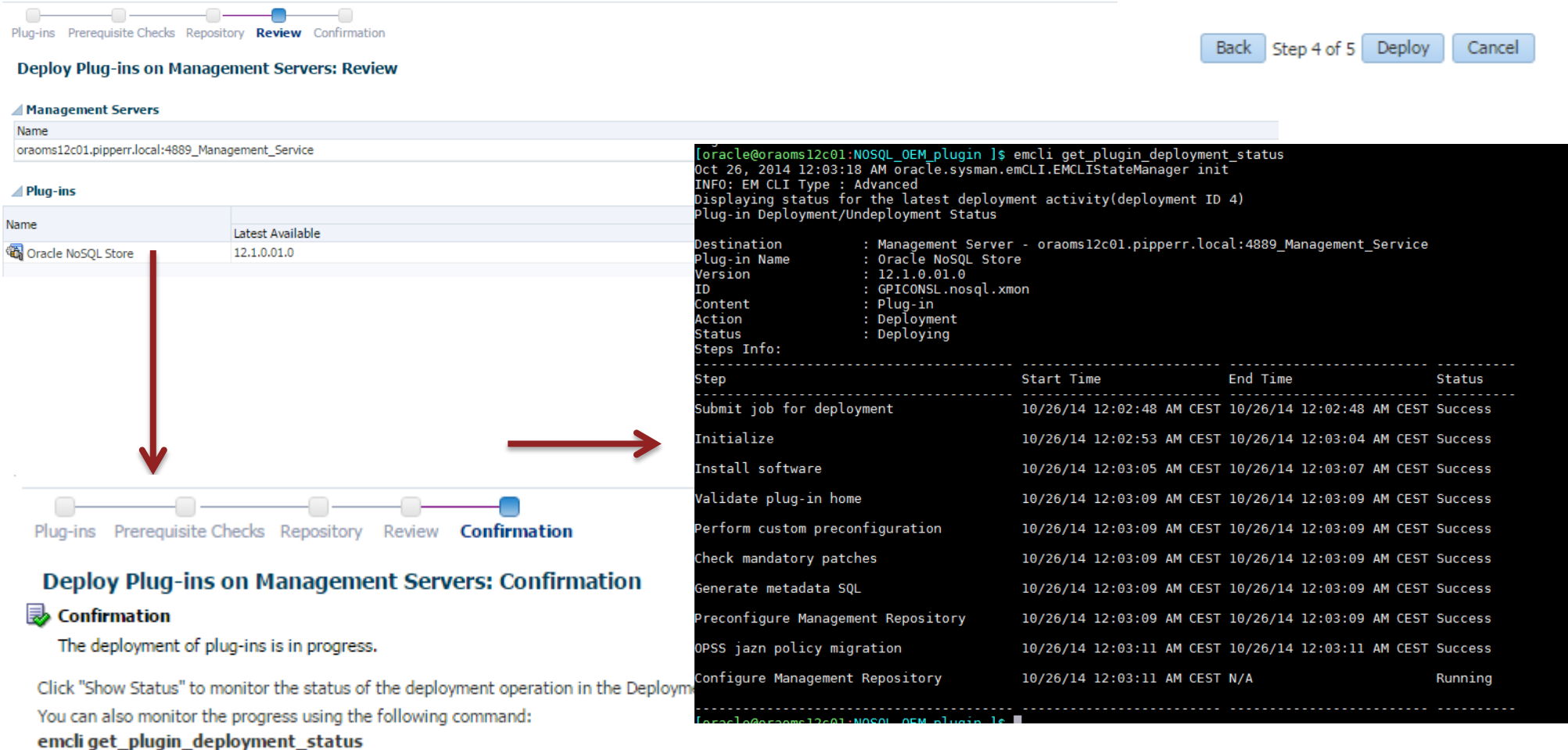
* Confirm Password

Role

✓ Save As

Plug-In auf dem OMS und dem Agent "ausrollen,, - 4

■ Plug-In auf dem OMS deployen



Plug-ins Prerequisite Checks Repository **Review** Confirmation

Deploy Plug-ins on Management Servers: Review

Management Servers

Name
oraoms12c01.pipperr.local:4889_Management_Service

Plug-ins

Name	Latest Available
Oracle NoSQL Store	12.1.0.01.0

```
[oracle@oraoms12c01:NOSQL_OEM_plugin ]$ emcli get_plugin_deployment_status
Oct 26, 2014 12:03:18 AM oracle.sysman.emCLI.EMCLIStateManager init
INFO: EM CLI Type : Advanced
Displaying status for the latest deployment activity(deployment ID 4)
Plug-in Deployment/Undeployment Status

Destination           : Management Server - oraoms12c01.pipperr.local:4889_Management_Service
Plug-in Name          : Oracle NoSQL Store
Version               : 12.1.0.01.0
ID                   : GPICONSL.nosql.xmon
Content               : Plug-in
Action                : Deployment
Status                : Deploying
Steps Info:
-----
Step                    Start Time                End Time                Status
-----
Submit job for deployment 10/26/14 12:02:48 AM CEST 10/26/14 12:02:48 AM CEST Success
Initialize              10/26/14 12:02:53 AM CEST 10/26/14 12:03:04 AM CEST Success
Install software         10/26/14 12:03:05 AM CEST 10/26/14 12:03:07 AM CEST Success
Validate plug-in home    10/26/14 12:03:09 AM CEST 10/26/14 12:03:09 AM CEST Success
Perform custom preconfiguration 10/26/14 12:03:09 AM CEST 10/26/14 12:03:09 AM CEST Success
Check mandatory patches 10/26/14 12:03:09 AM CEST 10/26/14 12:03:09 AM CEST Success
Generate metadata SQL    10/26/14 12:03:09 AM CEST 10/26/14 12:03:09 AM CEST Success
Preconfigure Management Repository 10/26/14 12:03:09 AM CEST 10/26/14 12:03:09 AM CEST Success
OPSS jazn policy migration 10/26/14 12:03:11 AM CEST 10/26/14 12:03:11 AM CEST Success
Configure Management Repository 10/26/14 12:03:11 AM CEST N/A Running
-----
```

Plug-ins Prerequisite Checks Repository Review **Confirmation**

Deploy Plug-ins on Management Servers: Confirmation

Confirmation

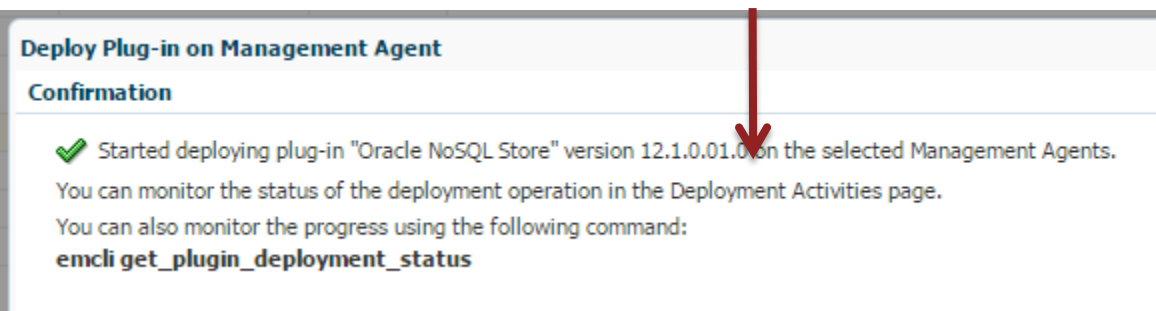
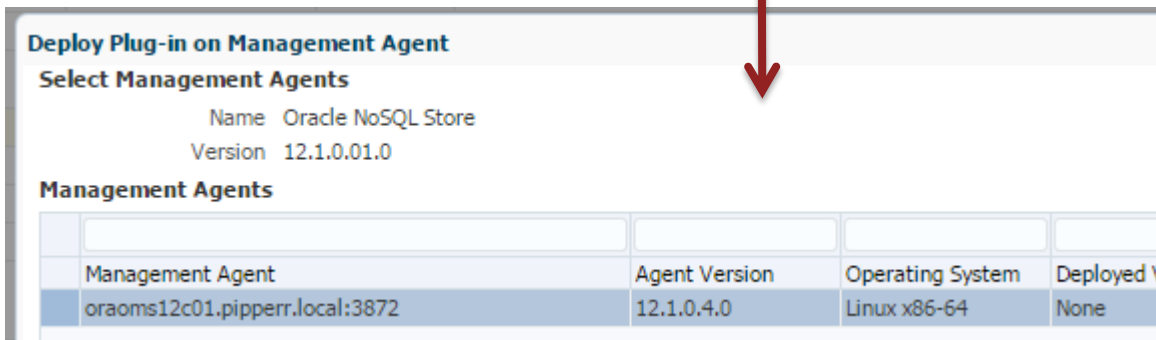
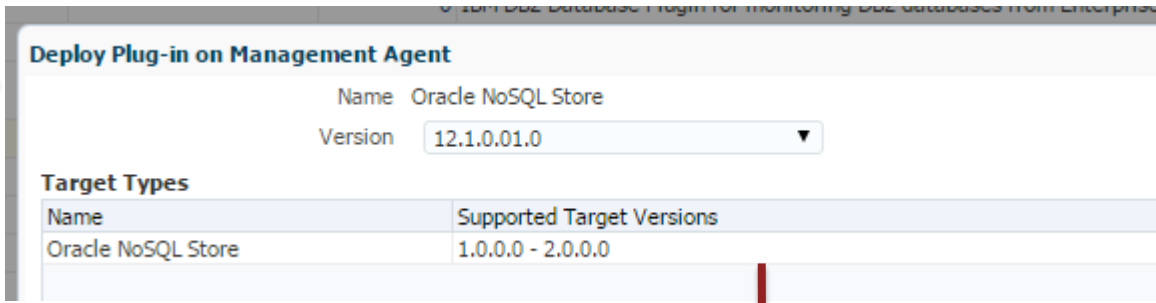
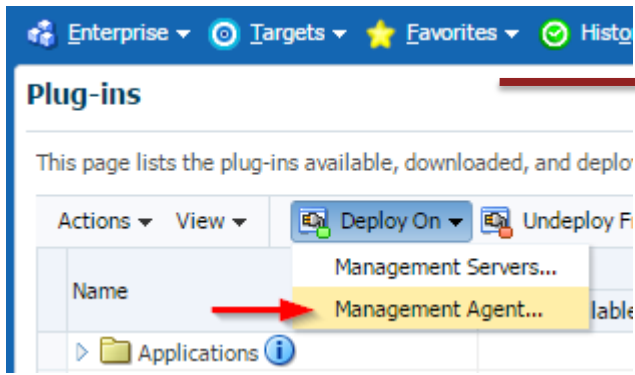
The deployment of plug-ins is in progress.

Click "Show Status" to monitor the status of the deployment operation in the Deployment page.

You can also monitor the progress using the following command:
`emcli get_plugin_deployment_status`

Plug-In auf dem Agent ausrollen

- Plug-In auf dem Agent ausrollen



Plug-In auf dem OMS und dem Agent "ausrollen"

- Plug-In auf dem Agent deployen
 - Target auswählen und dort das Plug-In deployen:

The screenshot displays the Oracle Enterprise Manager (OEM) interface. On the left, the navigation menu shows 'Add Targets Manually' selected. The main content area is titled 'Add Targets Manually' and includes an 'Instruction' section with three radio buttons: 'Add Host Targets', 'Add Targets Using Guided Process', and 'Add Targets Declaratively by Specifying Target Monitoring Properties' (which is selected). Below this, the 'Target Type' is set to 'Oracle NoSQL Store' and the 'Monitoring Agent' is 'oraoms12c01.pipperr.local:3872'. An 'Add Manually ...' button is visible. The 'Add Oracle NoSQL Store' form is shown below, with the following details:

Target

- * Target Name: GPI_STORE
- Target Type: Oracle NoSQL Store
- Agent: https://oraoms12c01.pipperr.local:3872/emd/main/

Properties

- * Communication Protocol: rmi
- * Host: oraoms12c01
- * JMX Port Number: 5000
- Main Directory for this store: /opt/oracle/kvdata1
- * Name of the KVStore: GPIDB
- * NoSQL Connection Port Number for first SN: 5000
- Oracle NoSQL Version: 12cR1.3.0.14
- * Service Name: jmxrmi

Global Properties

- Comment: [empty]
- Contact: [empty]
- Cost Center: [empty]
- Department: [empty]
- Lifecycle Status: Development
- Line of Business: [empty]
- Location: [empty]

A 'Confirmation' dialog box is shown at the bottom right, indicating that the target was added successfully:

Confirmation

Add Target - Completed Successfully

Hide

Added Oracle NoSQL Store GPI_STORE on https://oraoms12c01.pipperr.local:3872/emd/main/

Target aufrufen

ORACLE Enterprise Manager Cloud Control 12c

Enterprise Targets Favorites History

Search Target Name

All Targets Page Refreshed Nov 14, 2014 1:

Refine Search

Target Type Oracle nosql Store

Target Status Up (1)

Target Name	Target Type	Target Status
NoSQL_DCGPIDB_on_oraoms12c01.pipperr.local_5000	Oracle nosql Store	Up

ORACLE Enterprise Manager Cloud Control 12c

Enterprise Targets Favorites History

NoSQL_DCGPIDB_on_oraoms12c01.pipperr.local_5000

nosql Overview

Summary

Target Name: NoSQL_DCGPIDB_on_oraoms12c01.pipperr.lo
Target Type: nosql
Plug Version Nr.: 12.1.0.62.0
nosql Version Nr.: 12.1.3.0.14
Current Status: Up
Up Since: Nov 14, 2014 12:36 am CET
Hosted By: oraoms12c01.pipperr.local

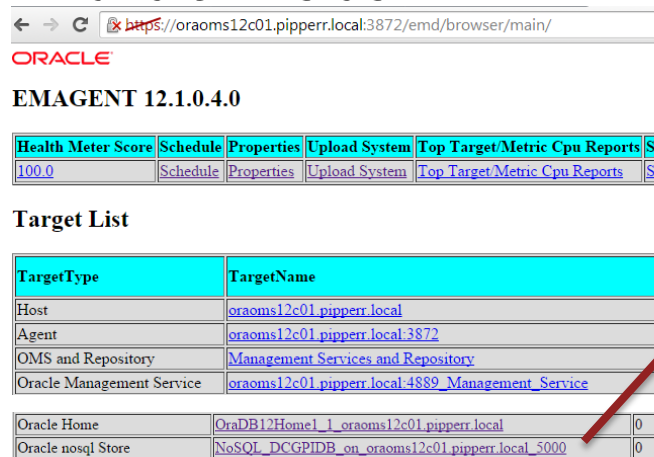
Storage Entries

SN or Rep Node	Details of the Node	Status ...	Master ...
sn1	[sn1] dc=dc1 oraoms12c01:5000	RUNNING	
[rg1-rn1] sn=sn1	[rg1]	RUNNING	MASTER
sn2	[sn2] dc=dc1 oraoms12c01:5002	RUNNING	
[rg2-rn1] sn=sn2	[rg2]	RUNNING	MASTER

Auf dem Node über die Agent Metriken die Metriken testen

- Über die URL des Management Agenten kann auf dem Node geprüft werden ob die Metriken erfasst werden können
- Voraussetzung : **Metrik Browser** auf dem Agent **aktiveren**
- Im Agent Home:
 - `emctl setproperty agent -name _enableMetricBrowser -value true`
 - `emctl reload`
 - URL <https://oraoms12c01.pipperr.local:3872/emd/browser/main> aufrufen und anmelden

als Agent OS User anmelden

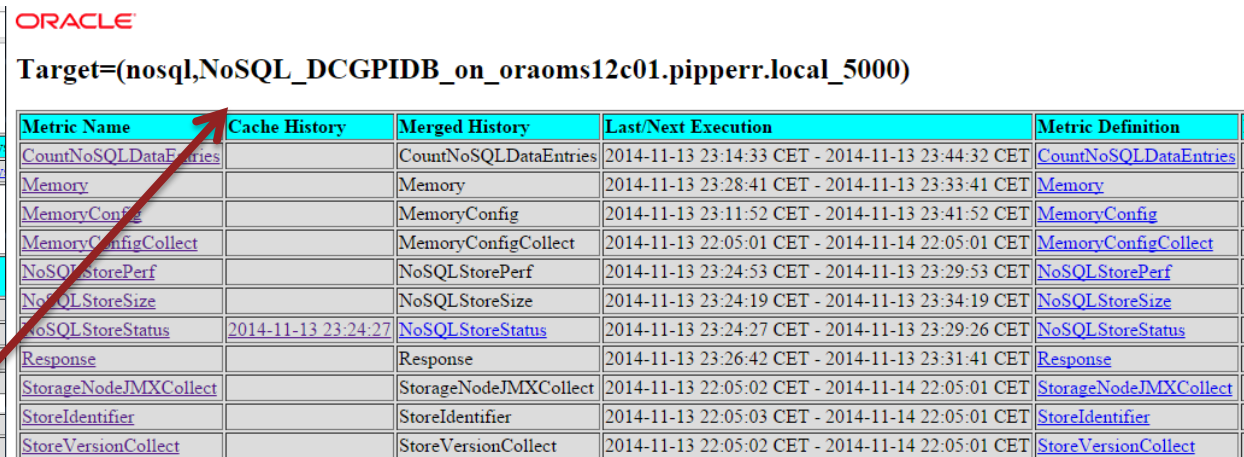


ORACLE
EMAGENT 12.1.0.4.0

Health Meter Score	Schedule	Properties	Upload System	Top Target/Metric Cpu Reports	Sys
100.0	Schedule	Properties	Upload System	Top Target/Metric Cpu Reports	Sys



Target List

TargetType	TargetName
Host	oraoms12c01.pipperr.local
Agent	oraoms12c01.pipperr.local.3872
OMS and Repository	Management Services and Repository
Oracle Management Service	oraoms12c01.pipperr.local.4889_Management_Service
Oracle Home	OraDB12Home1_1_oraoms12c01.pipperr.local 0
Oracle nosql Store	NoSQL_DCGPIDB_on_oraoms12c01.pipperr.local_5000 0



ORACLE
Target=(nosql,NoSQL_DCGPIDB_on_oraoms12c01.pipperr.local_5000)

Metric Name	Cache History	Merged History	Last/Next Execution	Metric Definition
CountNoSQLDataEntries		CountNoSQLDataEntries	2014-11-13 23:14:33 CET - 2014-11-13 23:44:32 CET	CountNoSQLDataEntries
Memory		Memory	2014-11-13 23:28:41 CET - 2014-11-13 23:33:41 CET	Memory
MemoryConfig		MemoryConfig	2014-11-13 23:11:52 CET - 2014-11-13 23:41:52 CET	MemoryConfig
MemoryConfigCollect		MemoryConfigCollect	2014-11-13 22:05:01 CET - 2014-11-14 22:05:01 CET	MemoryConfigCollect
NoSQLStorePerf		NoSQLStorePerf	2014-11-13 23:24:53 CET - 2014-11-13 23:29:53 CET	NoSQLStorePerf
NoSQLStoreSize		NoSQLStoreSize	2014-11-13 23:24:19 CET - 2014-11-13 23:34:19 CET	NoSQLStoreSize
NoSQLStoreStatus	2014-11-13 23:24:27	NoSQLStoreStatus	2014-11-13 23:24:27 CET - 2014-11-13 23:29:26 CET	NoSQLStoreStatus
Response		Response	2014-11-13 23:26:42 CET - 2014-11-13 23:31:41 CET	Response
StorageNodeJMXCollect		StorageNodeJMXCollect	2014-11-13 22:05:02 CET - 2014-11-14 22:05:01 CET	StorageNodeJMXCollect
StoreIdentifier		StoreIdentifier	2014-11-13 22:05:03 CET - 2014-11-14 22:05:01 CET	StoreIdentifier
StoreVersionCollect		StoreVersionCollect	2014-11-13 22:05:02 CET - 2014-11-14 22:05:01 CET	StoreVersionCollect



Die Version 2 deployen

Wie entwickle ich das nun weiter?

Eine neuere Version des Plug-Ins erstellen

- Für eine **neuere Version** muss die Plug-In Version **hochgezählt** und mit dem "create" Befehl ein neues Plug-In Bundle mit der neuen Versionsnummer erzeugt werden
- **ACHTUNG : !!BUG!!** (falls noch OEM Version 12.1.0.1.0 im Einsatz!)
 - Beim Erzeugen eines Installationspaketes werden in das „stage“ Verzeichnis unter agent/discovery/oms "install" Verzeichnisse angelegt
 - Diese Verzeichnisse müssen zuvor gelöscht werden, da ansonsten der Build abbricht

Fehler:

```
„2013-07-01 16:25:34,877 [main] ERROR packaging.PluginPackager logp.251  
– OSPOUIError oracle.sysman.pdk.packaging.OSPOUIError:  
Exit value of process oui/bin/runInstaller is non-zero: 255”
```

Zusammenfassung

- Bei jedem Full Deploy einer neuen Version Versionsnummer in "`plugin.xml`" und "`agent/plugin_registry.xml`" hochzählen
- **NIE** Versionsnummer **per Hand** auf dem **Agent höher als** auf dem **OMS** setzen (z.B. beim Testen auf dem Agent mit einer neueren `plugin.xml` Datei!)
 - Agent wird in den Status blocked versetzt und muss neu vom OMS aus synchronisiert werden!



Weitere Metriken im laufenden Betrieb entwickeln

Nach mehr Daten abfragen

Im laufende Betrieb Metriken deployen

- Im laufenden Betrieb können Änderungen an Metriken auch direkt deployed werden
- Vorteil: Es muss nicht jedes Mal bei jeder kleinen Änderung umständlich eine neue Version deployed werden
- 1. Deployment der Metrik Metadaten mit:
 - `${OMS_HOME}/bin/emctl register oms metadata -service targetType ...`
 - `${OMS_HOME}/bin/emctl register oms metadata -service storeTargetType ...`
 - `${OMS_HOME}/bin/emctl register oms metadata -service default_collection ...`
- 2. Kopieren der Metadaten per Hand in das Plug-In Verzeichnisse mit der richtigen Version vom Agent
- 3. Agent neu starten

Für diese Schritte empfiehlt es sich entsprechende Skripts zu erstellen!



Die OEM Homepage für das Target definieren

Eine hübsche Oberfläche erstellen

Eine eigene Home Page Seite definieren

- + ■ Varianten:
 - Default Seite des OMS für das Target verwenden (HTML)
 - Eine "angepasste" Seite mit einer XML Definition erstellen (FLASH)
 - Eine eigene FLASH Seite definieren

- Für das NoSQL Plug-In kommt die Variante 2 zum Einsatz
 - Über die Datei
„[oms/metadata/mpcui/metadata_ui_homepage_charts.xml](#)“
wird eine eigene Flash Seite deklarativ erstellt

OEM Page über die mpcui xml erzeugen

- Mit Hilfe der XML Definitionsdatei "oms/metadata/mpcui/metadata_ui_homepage_charts.xml" kann eine eigene Seiten im OMS definiert werden

- Ziel:

[NoSQL_DCGPIDB_on_oraoms12c01.pipperr.local_5000](#) oraoms12c01.pipperr.local
nosql Overview Page Refreshed Nov 14, 2014 12:49:58 AM CET

Summary

Target Name: NoSQL_DCGPIDB_on_oraoms12
 Target Type: nosql
 Plug Version Nr.: 12.1.0.62.0
 nosql Version Nr.: 12.1.3.0.14
 Current Status: Up
 Up Since: Nov 14, 2014 12:36 am CET
 Hosted By: oraoms12c01.pipperr.local

Storage Entries

SN or Rep N...	Details of the Node	Statu...	Maste...
sn1	[sn1] dc=dc1 oraoms12c01:5000	RUNNING	
[rg1-rn1] sn=sn1	[rg1]	RUNNING	MASTER
sn2	[sn2] dc=dc1 oraoms12c01:5002	RUNNING	
[rg2-rn1] sn=sn2	[rg2]	RUNNING	MASTER

Store Node Information

Datacenter Name: DCGPIDB
 Store Name: GPIDB
 Store Unique Name: STORE_localhost.localdomain
 Store Unique ID: 6c83c534-6cb9-48d3-ab14-433472be248c
 Root Directory:
 Num Partitions: 20
 Num Cpu Usage: NaN
 Cache Memory: NaN
 HeapMemory UsageMB: 13.603
 Non HeapMemory UsageMB: 12.615
 Total Memory UsageMB: 120.667

Storage Size on this node

166401.191 (Free Size MB inside the store disk (84.8%))
 2.733 (Size MB Used of the disk without the store (15.2%))
 29716.267 (Database Size (MB) on this Node (0%))

Storage Entries

Storage Operations per Minute

Total Operations per Minute (rg1-rn1) (blue line)
 Total Request per Minute (rg1-rn1) (orange line)

Incidents and Problems

* Target: Local target and related targets * Category: All

Summary	Target	Severity	Status	Escalation level	Type	Time Since Last Update

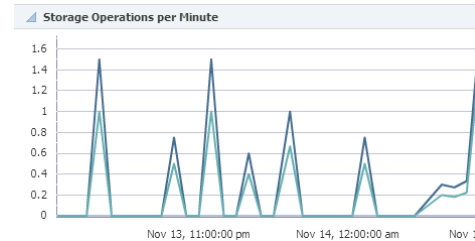
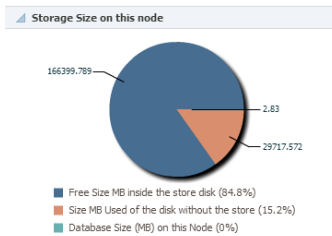
Mögliche Elemente

- Ein Metrik direkt in einer Tabelle anzeigen

Storage Entries

SN or Rep Node	Details of the Node	Status ...	Master ...
sn1	[sn1] dc=dc1 oraoms12c01:5000	RUNNING	
[rg1-rn1] sn=sn1	[rg1]	RUNNING	MASTER

- Ein Chart aus einer Metrik erzeugen



- Eine SQL Abfrage visualisieren

- Als Tabelle
- Als Chart

Store Node Information

Datacenter Name:	DCGPIDB
Store Name:	GPIDB
Store Unique Name:	STORE_localhost.localdomain
Store Unique ID:	6c83c534-6cb9-48d3-ab14-433472be248c

- „Incidents“ und „Problems“ darstellen

Incidents and Problems

* Target: Local target and related targets * Category: All | 0 | 0 | 0 |

Summary	Target	Severity

Target Home Page durch eigene Seite ersetzen

- Soll die Default Homepage im OEM mit einer eigenen Homepage Seite ersetzt werden:

- 1. Datei

"oms/metadata/systemUiIntegration/nosql_systemUiIntegration.xml" erstellen

```
<systemUiIntegration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.oracle.com/EnterpriseGridControl/SystemUiIntegration.xsd"
                    xmlns="http://www.oracle.com/EnterpriseGridControl/SystemUiIntegration">
  <general targetType="NoSQL"
            defaultLayout="twoColumnNarrowLeft"
            showOptionalRegions="true"
            topLevelTarget="true"
            allowCreateFromSystemsUi="true" />
</systemUiIntegration>
```

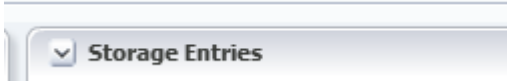
- In der "oms/mpcui/metadata_ui_homepage_charts.xml" eine Seite mit der ID "homePg" definieren

Danach muss das Plug-In als eine neue Version komplett auf dem OMS neu deployed werden, damit die Änderung auch wirklich verwendet wird

Fehlerquellen/Probleme bei der Entwicklung

- Bei Änderungen in der XML Datei sind bestimmte Änderungen nur durch ein **vollständiges neues Deployment** auf dem Management Server möglich:
 - Navigationselement in der Seite ändern / erstellen
 - System Home Page mit einer eigenen Seite ersetzen

Grundprinzip der „mpcui“ XML Datei

- Eine Seite (`mp:Page`) ist in Spalten (`mx:Vbox`) und Zeilen (`mx:Hbox`) aufgeteilt
- Elemente auf der Seite werden in Regions (`mp:Region`) "gruppiert"
 - Region ist die aufklappbare Einheit auf der Seite wie: 
- Elemente auf der Seite werden in XML definiert (wie `mp:Table`, `mp:chart`) und verwenden Datenquellen wie `mp:SQLDataService`
- Datenquellen werden pro Seite deklariert (`mp:services`)
- Datenquellen können global definierte SQL's verwenden (`SqlStatements`)
- Seiten und Dialog werden global registriert (`Integration`)

„mpcui“ XML – Elementstruktur muss definiert werden

- + ■ CustomUI
- SqlStatements
- UIMetadata
 - Integration
 - mp:Integration
 - » mp:PageActivityDef
 - » mp:DialogActivityDef
 - ActivityDefinition
 - » mp:Page
 - mp:services
 - mx:Hbox
 - mp:Region
 - mp:InfoDisplay oder mp:Table oder mp:LineChart etc.
- MenuMetadata
- EmuiConfig

Beispiel - metadata_ui_homepage_charts.xml (1)

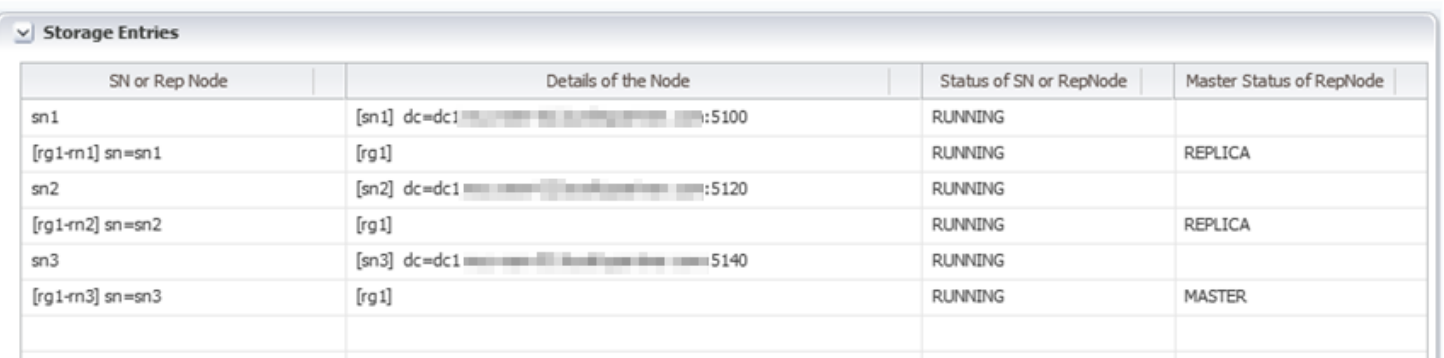
- Eine bestehende Metrik als Tabelle auslesen

```
<!-- Table View of one Metric -->
<mp:Region title="Storage Entries" width="75%" height="100%">

  <mp:Table id="NoSQLStatusOverview"
    width="100%"
    height="100%"
    metricName="NoSQLStoreStatus"
    metricColumns=" ['NODE', 'NODEInfo', 'NodeStatus', 'MasterStatus']"
    timePeriod="REALTIME"
    interval="30"

  <mp:columns>
    <mx:AdvancedDataGridColumn width="90" dataField="NODE" />
    <mx:AdvancedDataGridColumn width="160" dataField="NODEInfo" />
    <mx:AdvancedDataGridColumn width="70" dataField="NodeStatus" />
    <mx:AdvancedDataGridColumn width="70" dataField="MasterStatus" />
  </mp:columns>
</mp:Table>
</mp:Region>
```

Verweis auf die Metrik



SN or Rep Node	Details of the Node	Status of SN or RepNode	Master Status of RepNode
sn1	[sn1] dc=dc1 5100	RUNNING	
[rg1-rn1] sn=sn1	[rg1]	RUNNING	REPLICA
sn2	[sn2] dc=dc1 5120	RUNNING	
[rg1-rn2] sn=sn2	[rg1]	RUNNING	REPLICA
sn3	[sn3] dc=dc1 5140	RUNNING	
[rg1-rn3] sn=sn3	[rg1]	RUNNING	MASTER

Beispiel - metadata_ui_homepage_charts.xml (2)

- Ergebnis eines SQL's auf der Seite anzeigen
 - Zum Beispiel die Version des Stores

- SQL Definieren unter <SqlStatements>

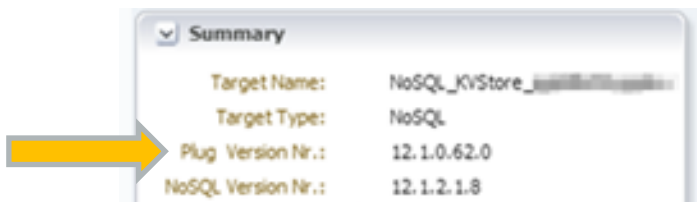
```
<Sql name="NOSQL_STORE_VERSION"><![CDATA[  
    SELECT ci.versionnr as "VersionNr"
```

- Service definieren unter <mp:services>

```
<mp:SQLDataService id="verInfo" queryID="NOSQL_STORE_VERSION" properties="{props('TARGET_GUID', appModel.target.guid)}" />
```

- Daten auslesen und in einem mp:InfoDisplay anzeigen

```
<mp:InfoItem label="NoSQL Version Nr.:" value="{verInfo.result.getString('', 'VersionNr')}" />
```



Beispiel - metadata_ui_homepage_charts.xml (3)

■ Existierende „Regions“ einbinden

– Job Activities

- `<mp:JobSummaryRegion width="25%" height="100%" />`

– Job Overviews

- `<mp:JobsActivityRegion id="jobsOverview" height="40%" />`

– status

- `<mp:StatusOverviewRegion id="statusOverview" height="50%" />`

– Issues overview

- `<mp:IssuesOverviewRegion id="issuesOverview" height="50%" />`

– IncidentRegion

- `<mp:IncidentRegion width="50%" height="100%" />`



Eigene Home Page debuggen

- Mit dem Anfügen von folgenden Parametern an die Url im OMS **"&loglevel=ALL,CONSOLE"** kann die Homepage Konsole geöffnet werden

↑ NoSQL_DCGPIDB_on_oraoms12c01.pipperr.local_5000 ⓘ
nosql Overview ▾ oraoms12c01.pipperr.local
Page Refreshed Nov 14, 2014 12:49:58 AM CET ↻

Summary

Target Name: NoSQL_DCGPIDB_on_oraoms1
Target Type: nosql
Plug Version Nr.: 12.1.0.62.0
nosql Version Nr.: 12.1.3.0.14
Current Status: ↑ Up
Up Since: Nov 14, 2014 12:36 am CET
Hosted By: oraoms12c01.pipperr.local

Storage Entries

SN or Rep Node	Details of the Node	Statu...	Mast...
sn1	[sn1] dc=dc1 oraoms12c01:5000	RUNNING	
[rg1-rn1] sn=sn1	[rg1]	RUNNING	MASTER
sn2	[sn2] dc=dc1 oraoms12c01:5002	RUNNING	
[rg2-rn1] sn=sn2	[rg2]	RUNNING	MASTER

Store Node Information

Datcenter Name: DCGPIDB
Store Name: GPIDB
Store Unique Name: STORE_localhost.localdomain
Store Unique ID: 6c83c534-6cb9-48d3-ab14-433472be248c
Root Directory:
Num Partitions: 20
Num Cpu Usage: NaN

Storage Size on this node

166399.645
2.803
29717.744

- Free Size MB inside the store disk (84.8%)
- Size MB Used of the disk without the store (15.2%)
- Database Size (MB) on this Node (0%)

Storage Entries

Storage Operations per Minute

Metric History

.....

Incidents and Problems

* Target: Local target and related targets * Category: All | 0 | 0 | 0 | 0

Summary	Target	Severity	Status	Escalation level	Type	Time Since Last Update
---------	--------	----------	--------	------------------	------	------------------------

```
2014-11-16 20:05:53.095 [MpCui] DEBUG Meta-Data Parser: parsing = {appModel.target.type}
2014-11-16 20:05:53.104 [MpCui] DEBUG BindingContainer.addBinding: [object MetricValuesDataService].targetType={appModel.target.type}
2014-11-16 20:05:53.110 [MpCui] DEBUG [object MetricValuesDataService] attribute: metricName MemoryConfig MemoryConfig
2014-11-16 20:05:53.119 [MpCui] DEBUG [object MetricValuesDataService] attribute: columns {[InitHeapMemoryUsageMB], 'MaxHeapMemoryUsageMB', 'UsedHeapMemoryUsageMB'} {[InitHeapMemoryUsageMB', 'Max
2014-11-16 20:05:53.126 [MpCui] DEBUG Meta-Data Parser: parsing = {[InitHeapMemoryUsageMB', 'MaxHeapMemoryUsageMB', 'UsedHeapMemoryUsageMB']}
2014-11-16 20:05:53.135 [MpCui] DEBUG [object MetricValuesDataService] attribute: timePeriod LAST_DAY LAST_DAY
```

Zusammenfassung

- + ■ Homepage wird über die mpcui XML Datei unter [oms/mpcui/xxxx.xml](#) definiert

- Soll die Standard Home Page gesetzt werden muss die Datei ["oms/metadata/systemUiIntegration/nosql_systemUiIntegration.xml"](#) angelegt und eine neue Version deployed werden!

- Eine eigene Homepage kann Daten aus SQL Abfragen oder integrierten Metriken darstellen
 - Charts und Tabellen möglich
 - Vordefinierte Regions für Status und Jobs
 - Es können Detail Charts als Dialoge geöffnet werden





Nächste Schritte

Weitere Features erstellen

Weitere Möglichkeiten realisieren

- +
- Collection Items definieren
 - Stammdaten des Targets einsammeln
- Discovery definieren
 - Automatisch erkennen, ob das Target auf dem Host existiert
- Topologien erkennen
 - Zusammenhänge zwischen den Targets im Repository hinterlegen





Zusammenfassung

Plug-In Development

- + +
 - Ein Plug-In wird deklarativ über XML Dateien entwickelt
 - Die eigentliche Kernfunktionalität – das Ermitteln der eigentlichen Daten erfolgt je nach Bedarf über **FETCHLETS**
=> Programmbausteine über Perl/Bash/Java Implementierungen
 - Drei Bereiche
 - Agent
 - OMS
 - Discovery

Fazit

- Mit dem Plug-In Konzept hat Oracle dem OEM in Richtung Kundenfreundlichkeit deutlich geöffnet
- Der Kunde hat nun die Möglichkeit beliebige Applikationen und Business Prozesse auch im Enterprise Manager mit zu verwalten
- Eine eigne Lizenz ist dazu nicht notwendig, allerdings ist im Detail zu prüfen, ob der Agent auf „nicht Oracle“ Hosts entsprechend lizenziert werden muss
- Leider wird noch nicht ein „Kunden Cluster Target“ unterstützt, auch das Verknüpfen von Targets über die Topologien könnte etwas besser dokumentiert werden
- Die Entwicklung selber ist aber mit Basis Kenntnissen in der Skript Programmierung gut durchführbar
- Die Dokumentation ist leidlich gut und mit den Beispielen der mitgelieferten Plug-Ins können zügig erste Erfolge erzielt werden

F *Fragen* *&* *A*



Quellen

Wo finde ich mehr zum Thema?

- ✦ ■ Oracle Online Dokumentation
 - http://docs.oracle.com/cd/E24628_01/doc.121/e25161/toc.htm

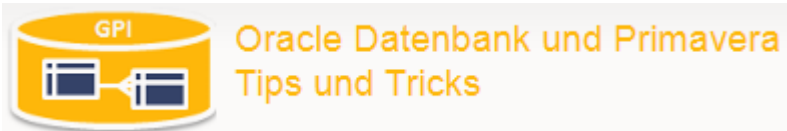
- Dokumentation im EDK Home unter
 - [doc/overview.html](#)
- Im "samples" Ordner des EDK
 - Beispiel Plug-Ins im EDK
 - Beispiel Homepages in den Zip Files

- Source Code der default Plug-Ins auf dem Installation Medium des OMS's auf der Disk 3

Quellen

- Mehr über das Thema siehe auch:

– http://www.pipperr.de/dokuwiki/doku.php?id=dba:oms_12c_plugin_development



Gunther Pippèrr - IT-Architekt - Berater



Background

Gunther Pippèrr arbeitet seit mehr als 16 Jahre intensiv mit den Produkten der Firma Oracle im Bereich Datenbanken/Applikationsserver und Dokumenten-Management.

Herr Pippèrr hat sich tiefes Wissen über den Aufbau komplexer IT Architektur aneignen können und hat dieses in der Praxis erfolgreich umgesetzt.

Herr Pippèrr hat eine Abschluss als Dipl. Ing. Technische Informatik (FH) an der FH Weingarten.

Functional Expertise

- IT System Architekt
- Technische Projektleitung
- Design und Implementierung von Datenbank Anwendungen
- Entwurf und Umsetzung von IT Infrastrukturen zum Datenmanagement

Industry Expertise

- High-Tech
- Real Estate
- Utility
- Communications

Web

<http://www.pipperr.de>
<http://orapowershell.codeplex.com>

Selected Experience

- Datenbank Architekt für ein Projekt zur Massendatenverarbeitung in der Telekommunikation
- Architekt und technische Projektverantwortung für ein Smart Metering Portal für das Erfassen von Energiezählerdaten und Asset Management
- Architekt und Projektleitung , Datenbank Design und Umsetzung für die Auftragsverwaltung mit Steuerung von externen Mitarbeitern für den Sprachdienstleister von deutschen Technologiekonzern
- Architekt und technische Projektverantwortung für IT Infrastrukturprojekte, z.B.:
 - Zentrale Datenhaltung für Münchner Hotelgruppe mit über 25 Hotels weltweit,
 - Redundante Cluster Datenbank Infrastrukturen für diverse größere Web Anwendungen wie Fondplattform und Versicherungsportale, Loyalty Card Anwendungen
- CRM- und Ausschreibungsportal für großen Münchner Bauträger