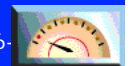


# SoDA Agent Customizations

---

- Module 1: Dashboard Overview
- Module 2: Dashboard Architecture
- Module 3: Dashboard Installation
- Module 4: Dashboard Operation and Customizations
- Module 5: SoDA / File Agent Basics
- **Module 6: SoDA Agent Customizations**
- Module 7: CSV Tool Basics and Operations
- Module 8: Dashboard API
- Module 9: Dashboard Administration



# Learning Objectives

---

- ↖ **When you complete this module, you should be able to**
  - Customize the SoDA agent initialization file to collect metric data from Rose, Requisite Pro, and ClearQuest.



# Rose 98i Domain Parameters Example

↖ *Collect a count of all Use Cases down to the package level*

Location of model file

inputFile=C:\Project\classics.mdl  
sodaDomain=rose98

Name the artifact,  
"UseCases"

Use Rose98 SoDA domain

hierarchyName=Classics  
hierarchyRoot=Classics:Rose  
ArtifactName=UseCases

Iterate over all "Packages". For each package, iterate over AllUseCases

Store in hierarchy "Classics"

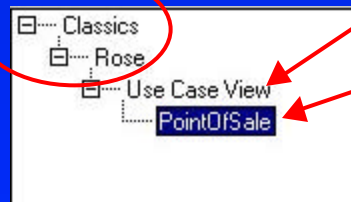
ArtifactSpec=Packages\*#:AllUseCases

Associate artifact with Package name

HierarchySpec=Name

ParentHierarchySpec=ParentPackage#:Name

Build hierarchy tree starting with Classics -> Rose ->



Fill in the hierarchy with the name of each of the use case's parent package

# Rose 98i Domain Parameters Example

↖ *Collect a count of all Classes down to the Class level*

inputFile=C:\Project\classics.mdl

sodaDomain=rose98

hierarchyName=Classics

hierarchyRoot=Classics:Rose

ArtifactName=UseCases

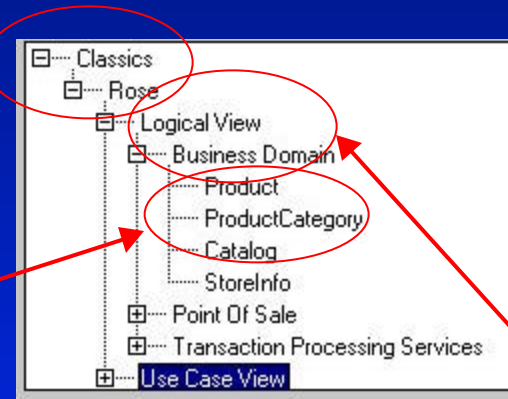
ArtifactSpec=AllClasses\*

HierarchySpec=Name

ParentHierarchySpec=ParentPackage#:Name

Associate artifact with Class names

Dynamically build the hierarchy using the name of the parent package(s)



NOTE: "ArtifactSpec=Package\*:AllClasses\*#" would double count for embedded packages (or more if you have multiply embedded packages)

# Rose 98i Domain Parameters Example

↖ *Collect a count of all Classes for only a specific Package*

inputFile=C:\Project\classics.mdl

sodaDomain=rose98

hierarchyName=Classics

hierarchyRoot=Classics:Rose

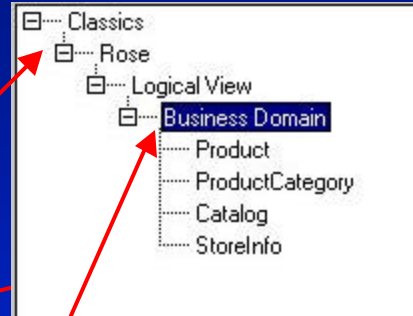
ArtifactName=Classes

ArtifactSpec=Packages\*:Name=Business

Domain:AllClasses\*#

HierarchySpec=Name

ParentHierarchySpec=ParentPackage#:Name



Name the artifact,  
"Classes"

Collect from  
AllClasses in Package  
"Business Domain"

Associate artifact  
with name of the  
Class



# Rose 98i Domain Parameters Example

↖ *Collect a count of all Classes with model property Instancing*

inputFile=C:\Project\classics.mdl

sodaDomain=rose98

hierarchyName=Classics

hierarchyRoot=Classics:Rose

ArtifactName=Instancing.

ArtifactSpec=AllClasses\*#:Properties\*:Name=Instancing:Value=Artifact  
Name

HierarchySpec=Name

ParentHierarchySpec=ParentPackage#:Name

Associate artifact  
with Class name

Name the artifact,  
"Instancing: <value>"

Collect from  
AllClasses with  
property Instancing

For each Instancing value,  
define artifact: E.g.,  
Instancing:Private,  
Instancing:PublicNotCreatable,  
Instancing:Multiuse, ...



# ReqPro Domain Parameters Example

## ↖ *Collect a count of all Requirements*

inputFile=C:\Project\classics.rqs

sodaDomain=ReqPro

hierarchyName=Classics

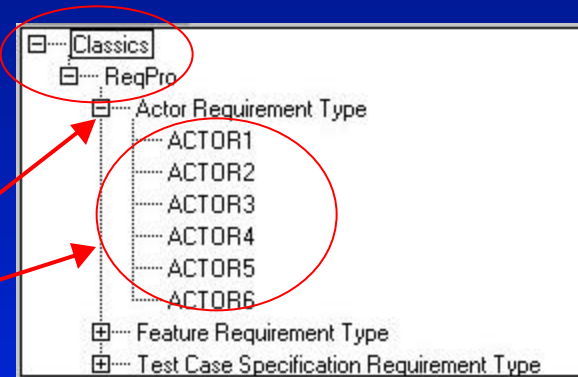
hierarchyRoot=Classics:ReqPro

ArtifactName=Requirements

ArtifactSpec=Requirements\*#

HierarchySpec=FullTag

ParentHierarchySpec=ReqType:Name



Associate artifact  
with Requirement

Name the artifact,  
Requirements



# CQ Domain Parameters Example

Collect a count of all defects based on the defect State field (values)

CQ Database name

inputFile=CLSIC  
username=alex  
password=alex

Username/ password to use

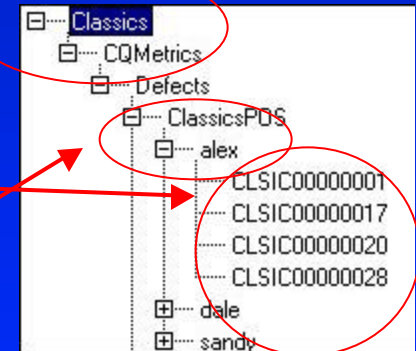
sodaDomain=ClearQuest  
hierarchyName=Classics  
ArtifactName=DefectState:

Name the artifact, "Defect:< state value>"

Create a hierarchy tree starting with Classics -> CQMetrics -> UserProblem

The "#" indicates that id (on the HierarchySpec query) is a query on a "Defect" record, not a "State" record

ArtifactSpec=Defect\*#:State=ArtifactName  
hierarchyRoot=Classics:CQMetrics:Defects  
HierarchySpec=id  
ParentHierarchySpec=owner:login\_name:@project:name



The "@" indicates that project:name is a query on a "Defect" record, not a "owner:login\_name" record



## Domain Parameters Detail

---

ArtifactSpec = AllUseCases\*#:Scenarios

### \* ⌞ Explanation

- *Collect a count of all Scenario or Interaction diagram for all the use cases specified in the Rose model*
- '\*' indicates that this is a repeating field (from SoDA domain)
  - e.g., AllUseCases\* and Scenarios\*
- '#' indicates that when building a hierarchy tree, this is the lowest level to be traverse, i.e., this becomes the leaf node
- Specify matching field



# SoDA Agent Customization Summary

- ↖ SoDA agent is customized using the INI file
- ↖ InputFile is either the data source (Rose MDL file, Requisite Pro RQS file) or the database name (ClearQuest)
- ↖ artifactSpec contains instructions for finding the artifacts
  - "\*" indicates that the field repeats
  - "#" points to the starting point for building the hierarchy
- ↖ hierarchySpec contains instructions for locating the "hierarchy" name for the artifact
- ↖ parentHierarchySpec contains instructions for building a hierarchy using multiple domain elements (either flat or hierarchical)
  - "#" indicates that this is a recursive query and sets the starting point for subsequent queries
  - "@" indicates that the next query should start at the "root"



## Lab 6: Dashboard SoDA Agent

---

- This exercise helps familiarize you with the Dashboard SoDA Agent customization – setup, execution, debugging
  - *Lab 6.1: Setup initialization file.*
  - *Lab 6.2: Collect Rose 98i data*
  - *Lab 6.3: Collect ReqPro data.*
  - *Lab 6.4: Collect ClearQuest data.*



## Lab 6.1: SoDA Agent Setup

---

- ↖ Initialization file located in `%SystemRoot%\SpcpSodaAgent.ini`
  - Edit the file and turn on debugging
    - `debug=true`
  - Run the agent "now"
    - `whenToRun=now`



## Lab 6.2: Collect Rose 98i Data

---

- ↖ **Set up the metric collection parameters to collect the number of operations for each class per package**
  - After setting up the parameters, run the SpcpSodaAgent interactively.
    - Notice the debug out.
  - Now launch the Dashboard client and see if the hierarchy was built correctly.
  - Verify that the operations were collected.
- ↖ **Extra credit: Set up the metrics collection parameters to collect all classes with "CollectionClass" model property set.**
  - Verify the collection.



## Lab 6.3: Collect ReqPro Data

---

- ▣ Set up the metrics collection parameters to collect a count of documents from the Classics demo.
  - Run the SpcpSodaAgent.
  - Verify the collection using the Dashboard client.



## Lab 6.4: Collect ClearQuest Data

---

- ▣ Set up the metrics collection parameters to collect a count of defects based on Severity in the Classics demo.
  - Run the SpcpSodaAgent.
  - Verify the collection using the Dashboard client.

[Next](#)

