TADDM v7.2.1
Grouping Composer

Document Version: 1.0

Document Status: Beta Draft

NOTE: The hard copy version of this document is FOR REFERENCE ONLY. Online version is the master

It is the responsibility of the user to ensure that they have the current version.

Any outdated hard copy is invalid and must be removed from possible use.

It is also the responsibility of the user to ensure the completeness of this document prior to use.

		IBM SWG - Tivoli Software	
	☎		
Issue Date:			
Authors:	**Scott W Graham** **TADDM Solutions Team**		

1 Document Control

1.1 Summary of Changes

The table below contains the summary of changes:

Version	Date	Description of changes
0.1	20.06.11	Initial Document creation
1.0 Beta	5.07.11	Beta Document distributed

2 Overview

IBM® Tivoli Application Discovery and Dependency Manager (TADDM): A robust application mapping and discovery tool that automatically gathers an inventory of all applications and dependencies, helps you understand configurations, and helps prove compliance. TADDM includes detailed reports and auditing tools.

TADDM 7.2.1 contains a number of key customer driven functional enhancements and market requirements including the following items:

- Usability, Visualization, and Time To Value
- Extending Portfolio Integrations
- Enhancing Virtualization Support
- Improved performance in the following areas
  - Sensor discovery
  - Bulk Loading
  - Web UI
  - API
  - Integration with other Products

Specific to this document, TADDM 7.2.1 provides a new tool called the 'Grouping Composer' that replaces Application Templates and the Collection Wizard. The purpose of this document is to help customers better understand what the Grouping Composer does and what problems it is best used to solve.

3 Prerequisites

The following documentation links in the InfoCenter provide some background information on using the Grouping Composer:

http://publib.boulder.ibm.com/infocenter/tivihelp/v46r1/topic/com.ibm.taddm.doc_721/UserGuide/c_em_groups.html

The Model Query Language:

http://publib.boulder.ibm.com/infocenter/tivihelp/v46r1/topic/com.ibm.taddm.doc_721/SDKDevGuide/c_cmdbsdk_mql_introducing.html

The Common Data Model (CDM):

http://publib.boulder.ibm.com/infocenter/tivihelp/v46r1/topic/com.ibm.taddm.doc_721/SDKDevGuide/c_cmdbsdk_understandingdatamodel.html

4 Grouping Composer

The Grouping Composer is a new TADDM tool to create *groups.* Groups within TADDM 7.2.1 consist of:

- **Business Applications** – A group of IT Resources that constitute a logical function or application. These are typically a set of servers and software programs that work together to provide a function to the business. For example, a typical three tiered web application would consist of an http server talking to an application server (Websphere) talking to a Database server.

- **Business Services** – A group of Business Applications that provide a larger service to the business itself. For example, an HR portal may contain a Payroll Business Application, a Directory Application and an Insurance Application. Collectively they are the HR Business Service.

- **Collections** – A group of resources that make sense to view or manage together. For example, all Linux Servers or all Database Servers.

- **Access Collections** – The same as Collections but can be used to restrict access via the UI and API to the resources they contain.

These groups can be **static, dynamic** or a **combination of both**.

Static Groups -- Groups that contain specifically selected assets from the TADDM Inventory.

Dynamic Groups -- Groups that use an MQL query to select assets. These groups' contents are regularly updated with the results of the queries.

Combination Groups – Groups created with MQL queries and statically defined assets.

The type of group to be created will largely depend on the desired contents. If the contents are well known and should not change significantly, or at all, over time then a **static group** may be the best choice. Conversely if the nature of the contents is dynamic in its own right, then a **dynamic group** would be the best choice. Most groups will probably end up as a **combination.**

4.1 Static Groups

Specifically selecting components to be part of a group is very useful in mocking up collections and applications or creating small groups that are not expected to change over time. In both of these cases, it is useful to hand pick the components of the group.

Creating a static group of a well-known application is the recommended first step to create a dynamic group. The static group will ensure that the correct dependencies are calculated and help determine what classes and queries would be necessary to create a dynamic group.

The primary advantage of a static group is that the learning curve to create one is much smaller than a dynamic group and does not require knowledge of MQL.

4.2 Dynamic Groups

The primary value of the grouping composer is creating 'dynamic groups' that will stay updated based on rules without user intervention. For example, to restrict the Linux Administrator team's access to just the Linux machines, a dynamic Access Collection could be created to contain all Computer Systems of type Linux. If new Linux computer systems were discovered, then they would be automatically added to the collection. Similarly, when systems were decommissioned they would be removed.

The challenge of creating a dynamic group is determining a pattern in the discovered data that can be accessed via MQL and will create the required group. If host and domain names contain a pattern that indicates a grouping, then that pattern can also be used to create the group. Any new machines created with that pattern would be automatically added to the group.

There are several reasons to choose a Dynamic Group:

1. The contents of a group is itself dynamic. For example, a business application may contain any number of http servers as they are added or removed for scaling purposes. In this case, the http tier contents is dynamic and a dynamic group using common attributes would be optimal to group the contents.

2. There are too many items to comfortably select for a static group. In some cases, it may be necessary to add all of a type of category to a group and while it will not be dynamic, it may be easiest to use an MQL query.

The general rule of thumb will be that if a pattern can be discerned via MQL, use a Dynamic Group. If it can't be, then use a Static Group.

4.3 Combination Groups

The most common type of group created in a production environment will be combination groups. Typical applications contain a selection of both static and dynamic content. For example a classic three tiered application may contain a variable number of Web Servers and Application servers but will only have one database over its lifetime. While the database could be specified using a dynamic MQL query, it is often faster to specify the database statically rather than determine the MQL necessary to determine the database.

When creating a combination group, the process will generally follow:

1. Create a Static Group to mock up the application as best possible.
2. Convert as many static components to common dynamic queries.
3. The remaining static components can remain static.

5 Tips and Tricks

The grouping composer is located in the 'Discovery Pane' in the Data Management Portal.

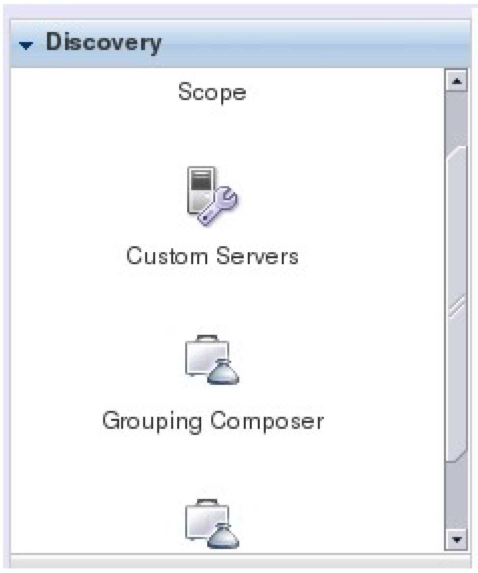


Illustration 1: Grouping Composer

After opening the Grouping Composer, press the 'New...' button to access the Grouping Composer Wizard. The wizard contains four panes to define the group.

General Information – Defines the basics of the group

Rules – Used to add dynamic content

Content – Used to add static content

Administration Information – Used to add administrator information about the group

5.1 General Information Window

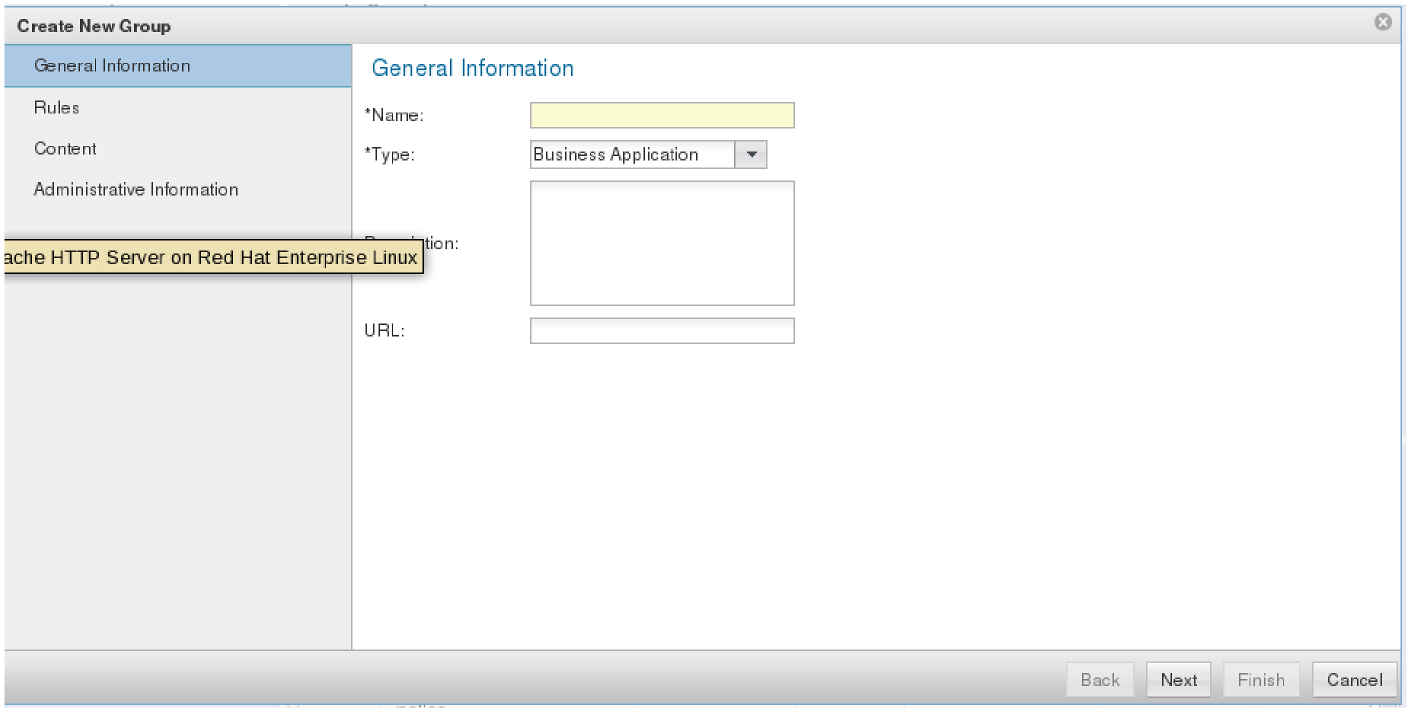


Illustration 2: General Business Application Pane

The GC 'General Information' window contains 4 fields that should be filled out when creating a Group.

5.1.1 Name

This will be the name of the 'Group'. If this group will be used in another product like TBSM, then pay careful attention to the characters used in the name. This is defined in the GC documentation here: http://publib.boulder.ibm.com/infocenter/tivihelp/v46r1/index.jsp?topic=/com.ibm.taddm.doc_721/UserGuide/c_em_groups.html

Specifically, you must not use the following characters in the group name:

- "
- <
- >
- \
- *
- ?
- |

- ;

5.1.2 Type

There are three types to choose from. These are defined in more detail in Section 4:

Business Service -- A collection of Business Applications

Business Applications – A collection of IT Resources that make up a logical application

Collection – A generic group of resources in TADDM. A collection can be used as an Access Collection to limit access to different resources.

Business Service and Business Applications also have two additional fields that collections do not have.

5.1.3 Description

The description is self explanatory but should be a brief summary of the application or service. This is not intended for Administration Information since there is an opportunity to provide that later. Details that are germane to administrators of the application or service and the TADDM administrators should be placed here. It is not necessary to place too much information here if there is URL that describes the application in detail. If there is a URL, place it in the URL field.

5.1.4 URL

This is a URL that will be accessible through the Business Application or Business Service for users to look up more information about the Object.

5.1.5 Collections

If a collection is chosen, then the only option will be to make the Collection an Access Collection. The only difference between the two is that an Access Collection, as the name describes, can be used to restrict access to the contents of the collection. A normal collection would be used to group similar contents and provide a shortcut to a group of items that an administrator cares about.

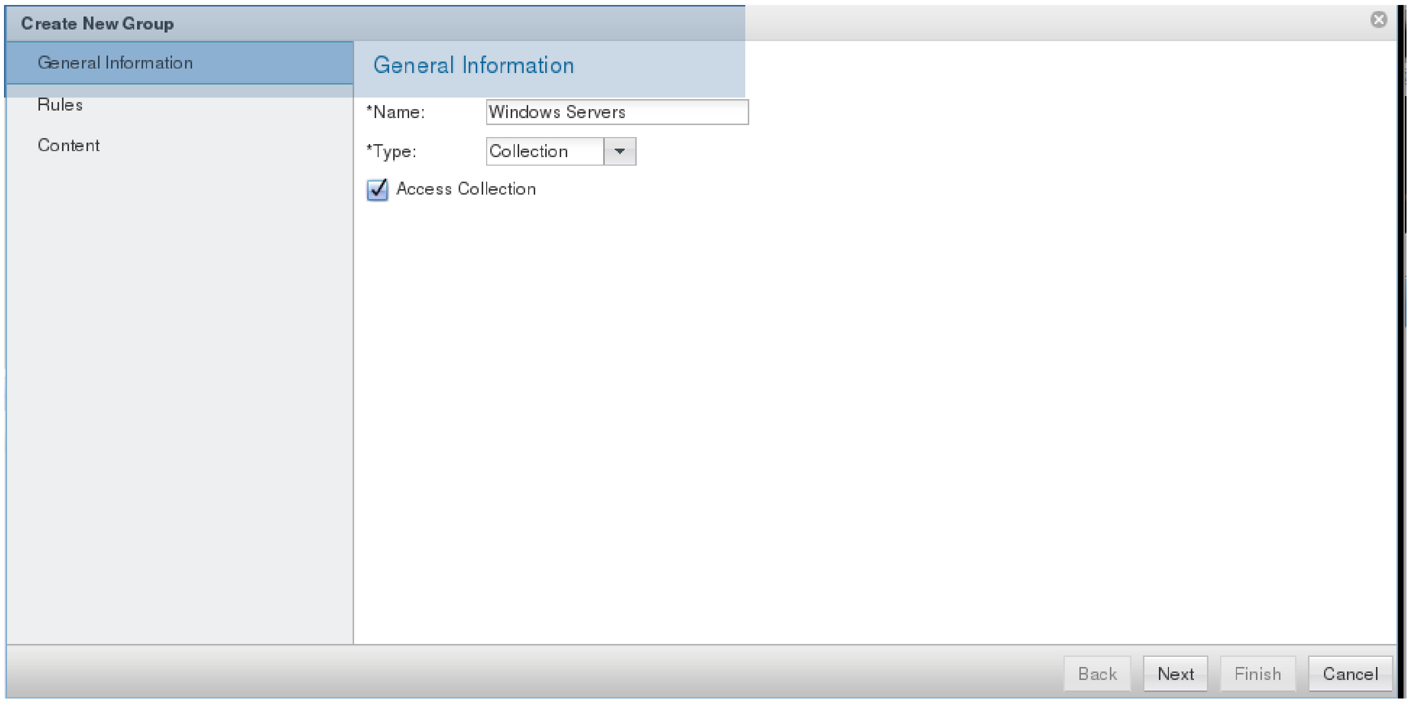

Create New Group
General Information
Rules
Content
General Information
*Name: Windows Servers
*Type: Collection
☑ Access Collection
Back Next Finish Cancel

Illustration 3: General Collections Pane

5.2 Rules

The rules page is the primary way to define **dynamic groups.** Rules are created using the Model Query Language (MQL) to select groups of TADDM objects that match a pattern. The language is similar to SQL but with a slightly different syntax intended to interact with Objects rather than strict tables. More information about MQL is available in the SDK guide here:

http://publib.boulder.ibm.com/infocenter/tivihelp/v46r1/topic/com.ibm.taddm.doc_721/SDKDevGuide/c_cmdbsdk_mql_introducing.html

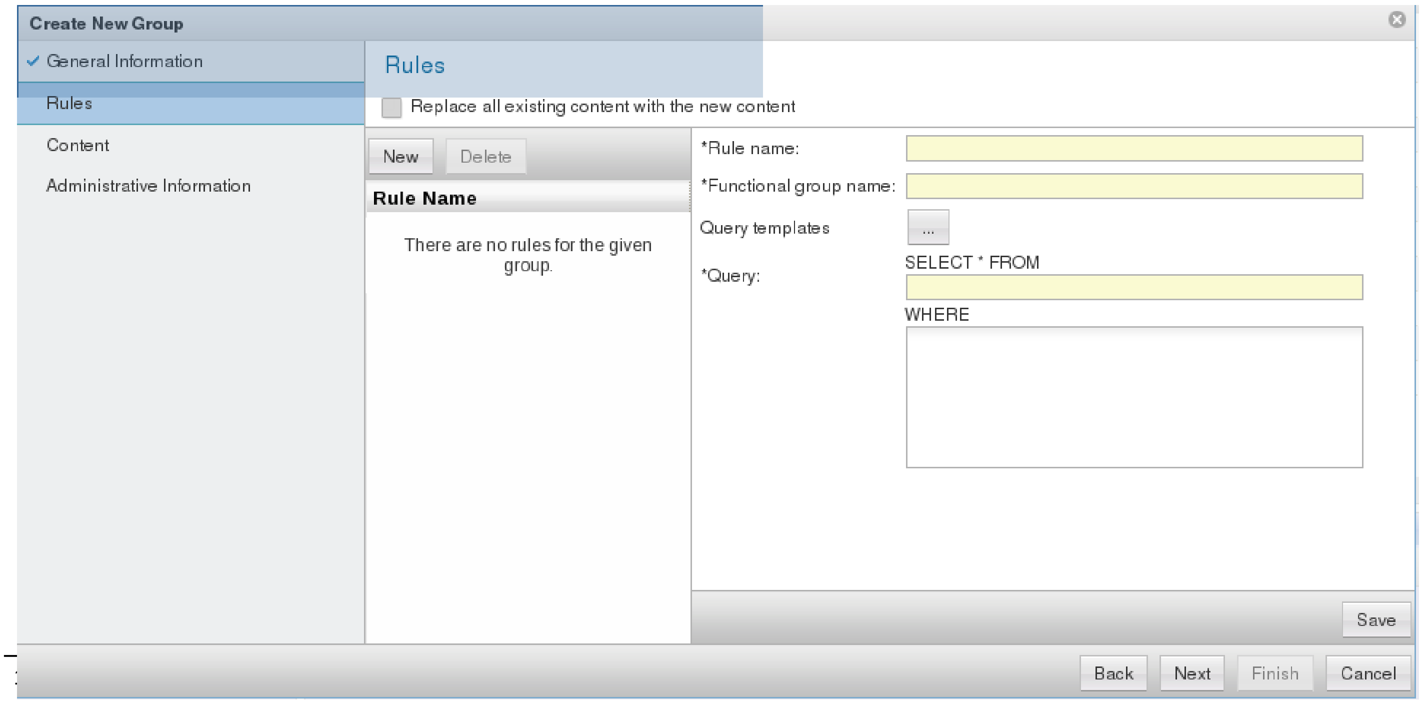

Create New Group
✓ General Information
Rules
Content
Administrative Information
Rules
☐ Replace all existing content with the new content
New Delete
Rule Name
There are no rules for the given group.
*Rule name:
*Functional group name:
Query templates ...
*Query: SELECT * FROM
WHERE
Save
Back Next Finish Cancel

Illustration 4: Rules Pane

TIP: When attempting to define a rule for a dynamic group, consider the variability of the contents. There are several reasons to use Static Content rather than a Dynamic Group:

1. If the contents will not change once defined, then it may be best to just use static content. For example, a database in a three tier application may never change and it would be more simple to define it statically.
2. If defining the contents is too difficult to do in MQL, it may be best to use static content.

The **Replace all existing content with the new content** checkbox indicates that the entire content of the Group will be updated with the new content from the rules for the group. Otherwise the new results will be added to the group. If you want items to be removed from groups when they no longer exist in TADDM (or match the query) then this option **must be selected.**

5.2.1 Rule Name

The Rule Name is only relevant to the current group. A rule will be used to create a dynamic collection of items from TADDM into the current group. Choose a name that indicates the objects that are expected to be returned. If they are ComputerSystems, then the rule could be named 'ComputerSystems'.

5.2.2 Query

As previously discussed, the Query option uses the Model Query Language (MQL). There are two fields that need to be filled out, the FROM and WHERE clauses. In the simplest form, the FROM clause would contain a Single Classname like 'ComputerSystem' and the WHERE clause would contain a filter on an attribute to select a subset of the ComputerSystem class.

FROM Clause

The from clause will be either a single ClassName or multiple ClassNames if the WHERE clause will contain a join. The Classnames can be found here:

`http://`*`taddmserverhost`*`:9430/cdm/datadictionary/model-object/index.html`

All of these classes will not work as they must be persistent. Whether a class is persistent is not obvious in this documentation and the easiest strategy is to attempt to do a 'select * from desirecclass' using the API. (See note below)

The most common Classnames to use will be:

ComputerSystem (or one of its children like: AixUnitaryComputerSystem, LinuxUnitaryComputerSystem, SunSPARCUnitaryComputerSystem, WindowsComputerSystem)

```
see: http://taddmserverhost:9430/cdm/datadictionary/cdm/classes/sys/ComputerSystem.htm
```

and

AppServer (or one of its children like: DatabaseServer or WebServer)

```
see:  http://taddmserverhost:9430/cdm/datadictionary/cdm/classes/app/AppServer.htm
```

NOTE: 'taddmserverhost' would be the hostname of your Primary Storage Server or the Domain Server in your environment.

WHERE Clause

The contents of the where clause can be used to filter the objects returned in the FROM clause. Typically, the filter will be used to describe like objects by determining a common attribute or **pattern** in an attribute. For example, to group ComputerSystems based on their domain, you would do:

```
“SELECT * FROM ComputerSystem WHERE fqdn ends-with 'sub.domain.com'”
```

All ComputerSystems with an fqdn of 'hostname.sub.domain.com' would be placed in the resulting Group.

To find all AppServers running on these ComputerSystems, the query would be:

```
“SELECT * FROM AppServer WHERE host.fqdn ends-with 'sub.domain.com'”
```

NOTE: The Query interface will not validate or test the query until the 'Finish' button is pressed. In order to preview the results of a Query or to test what is returned you should use the 'api.sh' command. Open a command line on the TADDM Server (Primary Storage Server). In the following example, COLLATION_HOME is where TADDM is installed. It is typically /opt/IBM/taddm/dist.

1. `cd $COLLATION_HOME`

2. `sdk/bin/api.sh -u <user_id> -p <password> find “SELECT * FROM <ClassName> WHERE <where-clause>”`

In the above, <user_id> is a TADDM ID that can login to the TADDM user interface and <password> is the ID's password.

<ClassName> and <where-clause> are the same as discussed above.

Text 1: Query Testing with API

5.2.3 Query Templates

The Query Template interface provides an interface to the 'Custom Query' wizard. This is discussed in the documentation here: http://publib.boulder.ibm.com/infocenter/tivihelp/v46r1/topic/com.ibm.taddm.doc_721/UserGuide/c_em_analytics_customquery.html

The contents of a custom query can be used to build a Rule. The Custom Query wizard lets the user build Queries without knowing ClassNames or attributes to filter on. The GUI walks the user through building these queries.

Two observations:

1. The Custom Query wizard only shows Class names for things that **have been discovered**. You cannot write a query that will return nothing.

2. The Custom Query wizard shows the child class (i.e. Linux) rather than 'ComputerSystem'. In order to create a query that returns all ComputerSystems, each type would have to be selected. To select a Parent class and its contents, use the normal Query.

5.2.4 Functional Groups

Functional Groups are primarily used to organize components to help application comparison. For example, when comparing a typical three tiered application with a Database, Web Server and Application Server each tier could be placed in a group a similarly named group.

Tip: In order to compare components in functional groups across business applications the fuctional groups should have the **same name.** For example, all databases in different business applications should be the same functional group if comparison is desired.

Generally speaking, place items in the group called 'Default Functional Group' unless a specific need presents itself. This will allow for comparison and simplify the requirement to match group names.

5.3 Content

The goal of a group or collection should be to automatically group a set of components based on a pattern. However if the pattern is too narrow (i.e. There is a single hostname and it will not change) then statically specifying the components via Content may be the simplest choice.

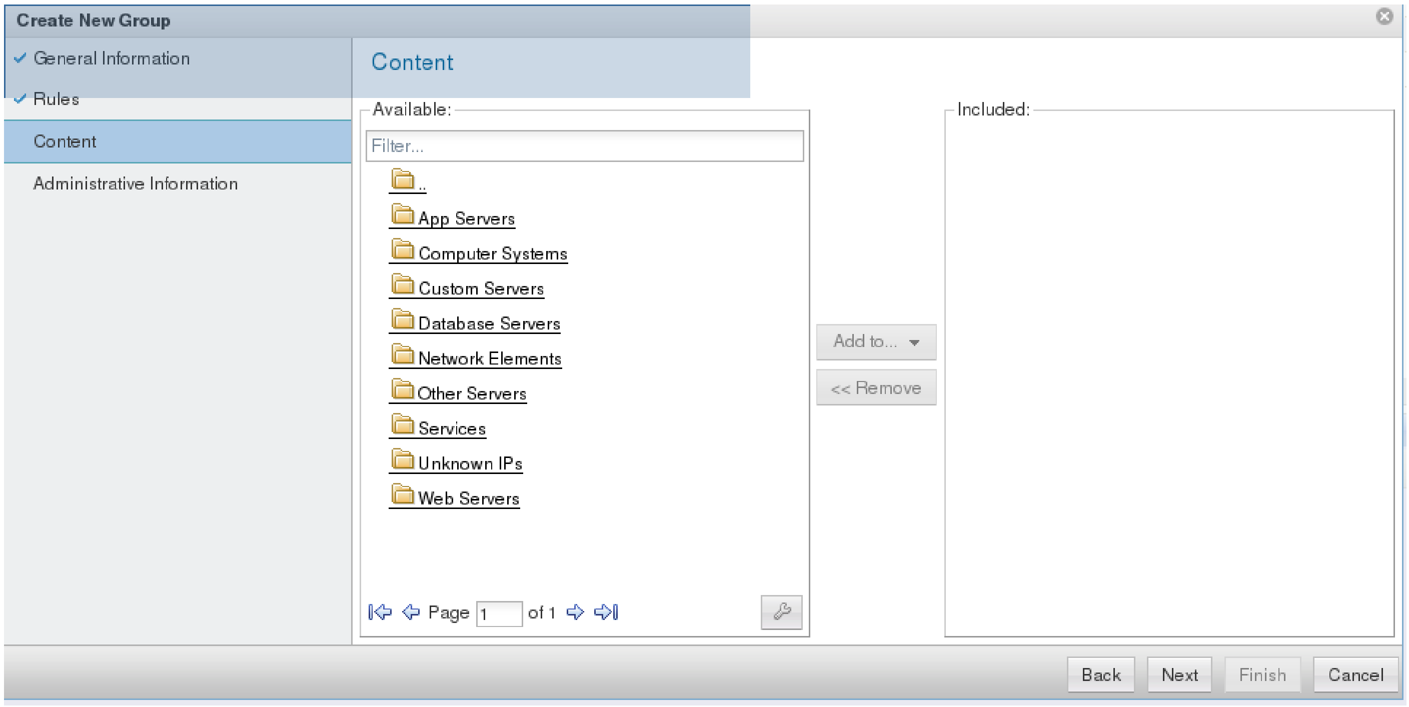


Illustration 5: Content Pane

Components specified in the Content Pane will be 'static components' and will always be used to build the collection.

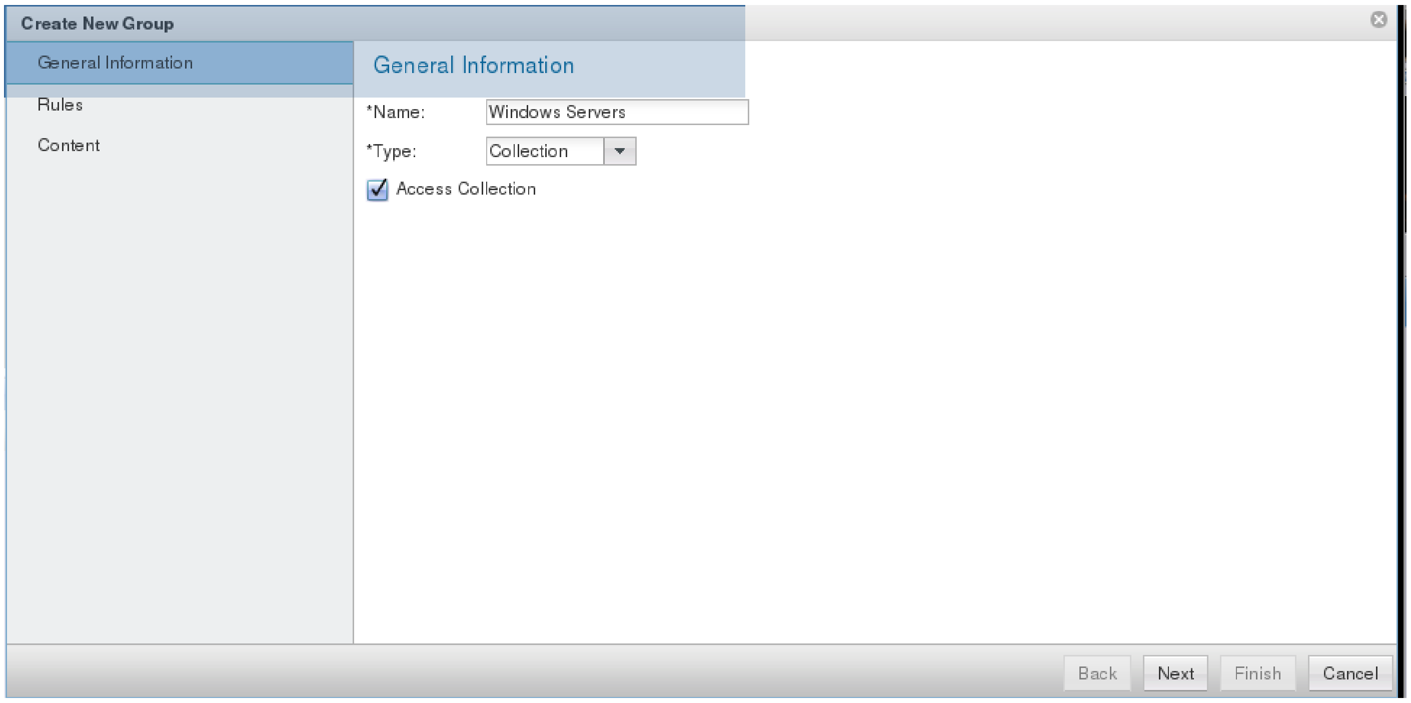
Tip: Use the Content pane to mock up an application when the contents are specifically known. This provides a simple way to create small collections and view the relationships between the components.

6 Examples

6.1 Creating Access Collections for all Unix and Windows Hardware

Problem: As an administrator of either the Unix or Windows environment at my company, I only need access to my machines and should not have access to the other machines.

Creating a group based on the OS is relatively simple. After launching the Grouping Composer and creating a New Group, do the following (in this case we are building the Windows Server collection):



*Name: Windows Servers
*Type: Collection
✓ Access Collection

Illustration 6: Windows Group

There are several ways to determine the Operating System of a Computer System. The class of a ComputerSystem usually indicates the operating system it is running. For example, a Windows Computer System class is 'WindowsComputerSystem' and a Linux Computer System class is 'LinuxUnitaryComputerSystem'. Alternatively, the OSRunning.OSName can be used to select ComputerSystems based on Operating System as well.

The following 'Rules' can be used to find the Windows Computer Systems:

```
SELECT * FROM ComputerSystem WHERE OSRunning.OSName contains 'Windows'
```

The 'contains' in the above query is necessary to get all Windows type machines as some may be 'Windows Server 2003' for example.

or

```
SELECT * FROM WindowsComputerSystem
```

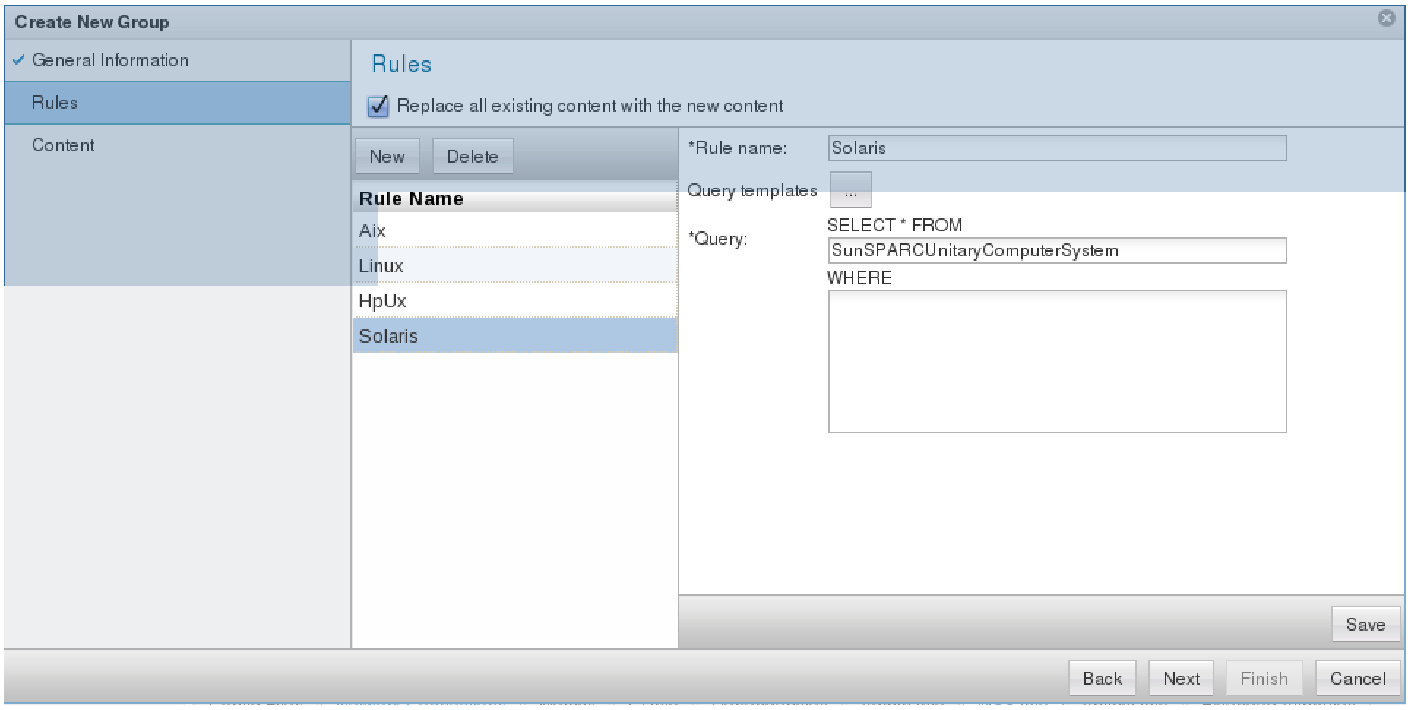
A Unix machine could be running any number of operating systems: Linux, HP-UX, Solaris or AIX. The rules to collect all Unix machines into a Group are more complicated:

```
SELECT * FROM ComputerSystem WHERE OSRunning.OSName IN ('Linux','HP-
UX','AIX','SunOS')
```

Or using a specific rule for **each** ComputerSystem type:

```
SELECT * FROM AixUnitaryComputerSystem
SELECT * FROM LinuxUnitaryComputerSystem
SELECT * FROM HpUxUnitaryComputerSystem
SELECT * FROM SunSPARCUnitaryComputerSystem
```

This would look like:



Replace all existing content with the new content ✓

Rule Name
Aix
Linux
HpUx
Solaris

*Rule name: Solaris

*Query: SELECT * FROM SunSPARCUnitaryComputerSystem WHERE

Illustration 7: Solaris Rule

6.2 Creating Access Collections based on Site

Problem: Create an Access Collection based on the populated Site Information to limit administrator access by site.

1. Populate the Administration Information tab of a ComputerSystem specifying the Site.

This is done **manually** by selecting the component in the 'Discovered Components' Pane

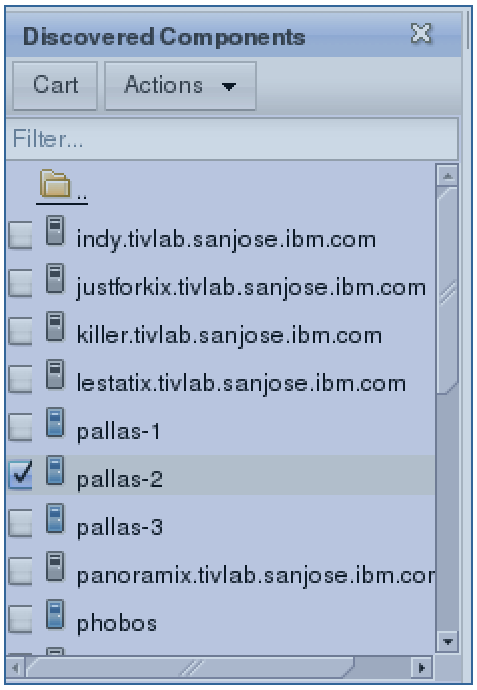
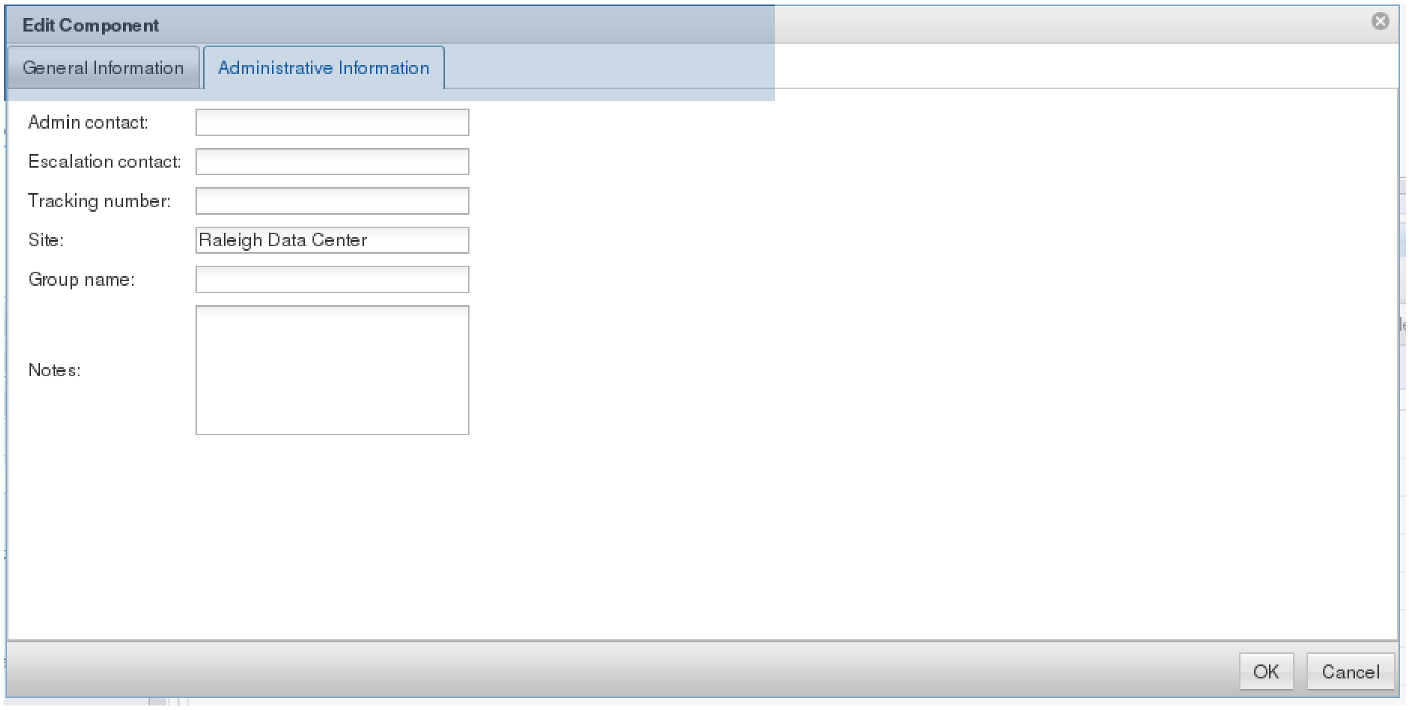

Discovered Components
Cart
Actions
Filter...
..
indy.tivlab.sanjose.ibm.com
justforkix.tivlab.sanjose.ibm.com
killer.tivlab.sanjose.ibm.com
lestatix.tivlab.sanjose.ibm.com
pallas-1
pallas-2
pallas-3
panoramix.tivlab.sanjose.ibm.cor
phobos

Illustration 8: Site

Then selecting Actions:Edit...



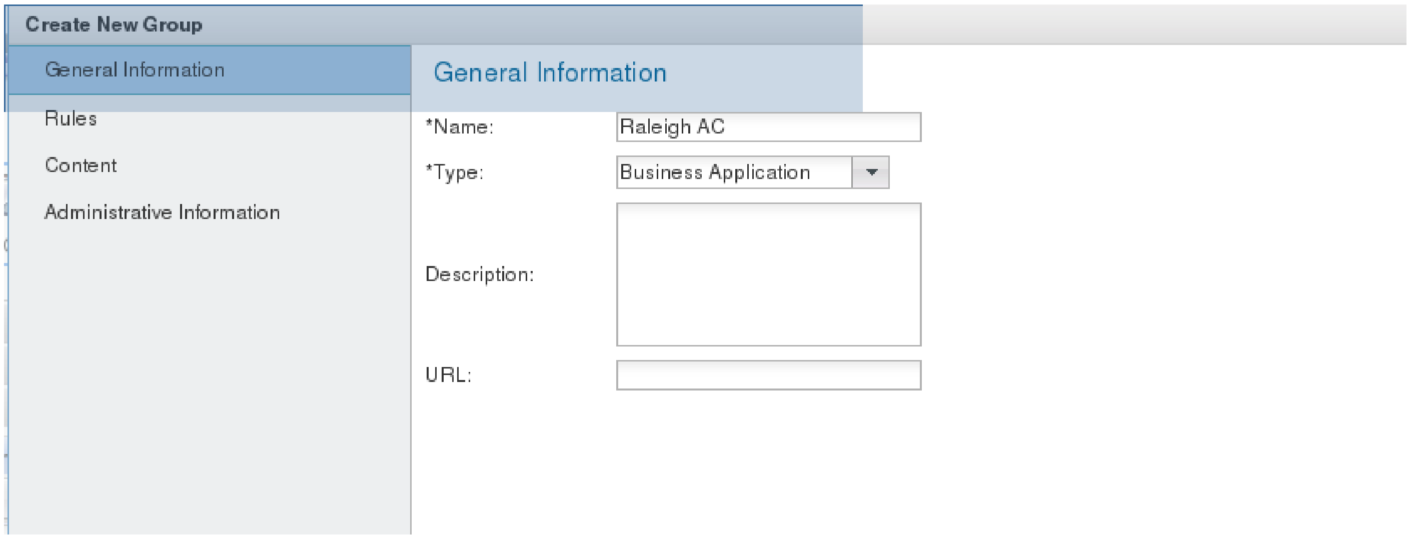
Admin contact:	
Escalation contact:	
Tracking number:	
Site:	Raleigh Data Center
Group name:	
Notes:	

Illustration 9: Raleigh Site Definition

Specify a Site name in the Site field. In this example, the Site is 'Raleigh Data Center'

NOTE: It is possible to automate this effort and this is discussed in detail on the TADDM Developerworks wiki but presently beyond the scope of this document.

2. Create an Access Collection.



*Name:	Raleigh AC
*Type:	Business Application
Description:	
URL:	

Illustration 10: Raleigh Definition

3. Specify the Rules for the Access Collection.

In order to allow access to all components that could be part of the ComputerSystem where the Site Information was displayed, Rules need to be created for:

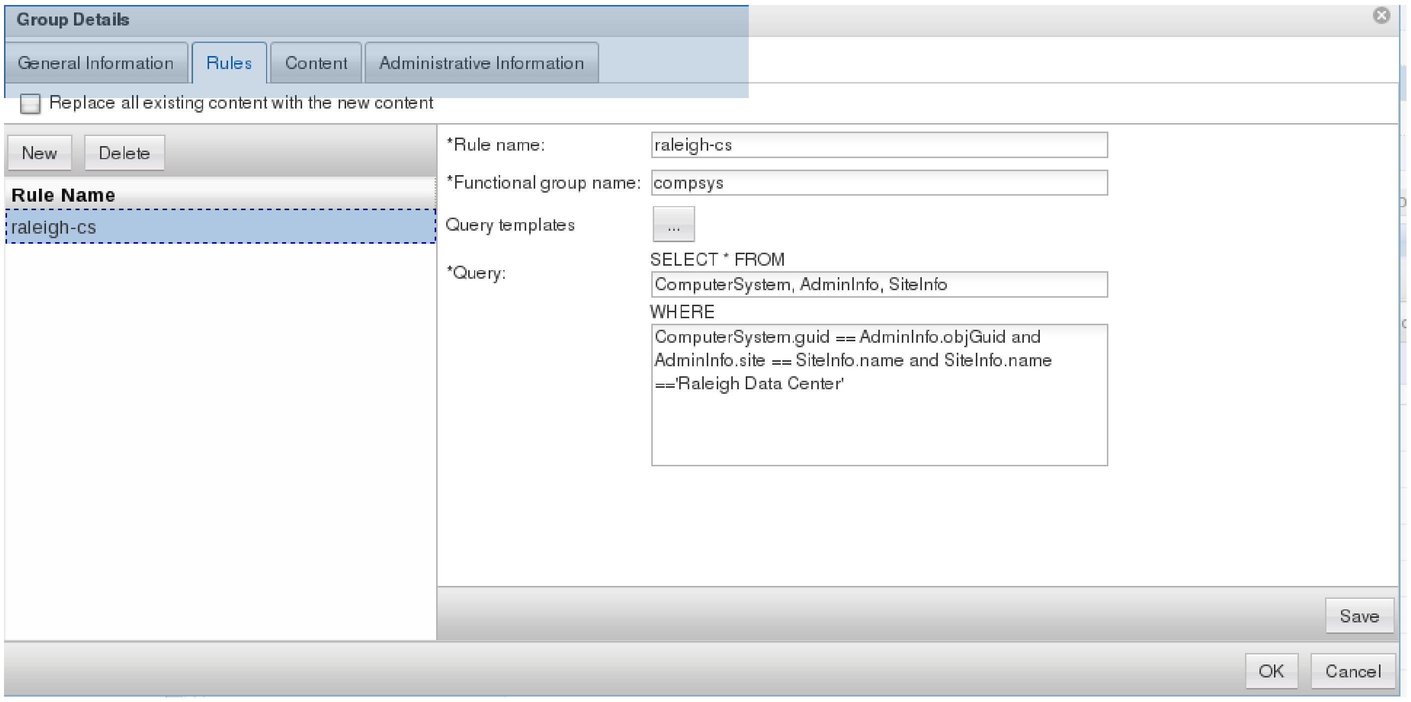
ComputerSystem

AppServer

Services

ComputerSystem

```
SELECT * from ComputerSystem, AdminInfo, SiteInfo WHERE (ComputerSystem.guid
== AdminInfo.objGuid and AdminInfo.site == SiteInfo.name and SiteInfo.name
=='Raleigh Data Center' )
```



Replace all existing content with the new content

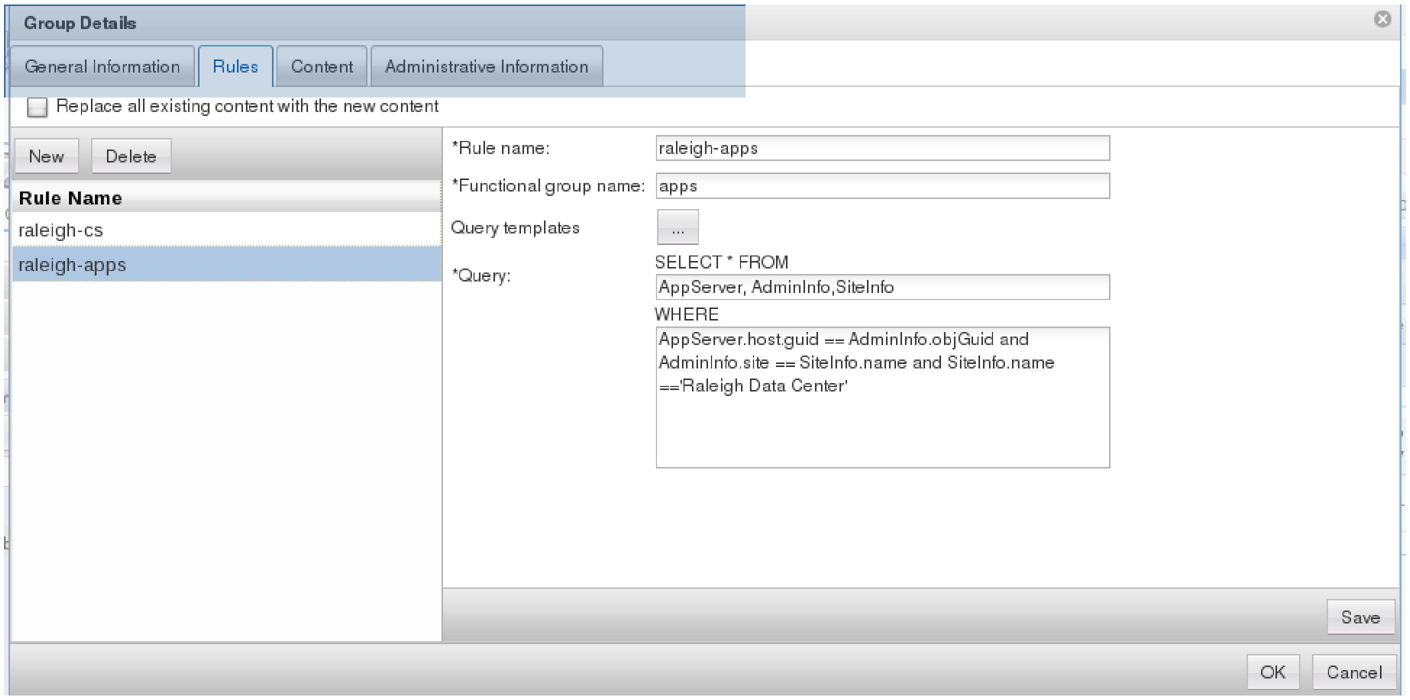
Rule Name
raleigh-cs

*Rule name: raleigh-cs
*Functional group name: compsys
Query templates ...
*Query:
SELECT * FROM
ComputerSystem, AdminInfo, SiteInfo
WHERE
ComputerSystem.guid == AdminInfo.objGuid and AdminInfo.site == SiteInfo.name and SiteInfo.name =='Raleigh Data Center'

Illustration 11: Raleigh ComputerSystem Rule

AppServer

```
SELECT * from AppServer, AdminInfo, SiteInfo WHERE (AppServer.host.guid ==
AdminInfo.objGuid and AdminInfo.site == SiteInfo.name and SiteInfo.name
=='Raleigh Data Center' )
```



Replace all existing content with the new content

Rule Name
raleigh-cs
raleigh-apps

*Rule name: raleigh-apps

*Functional group name: apps

*Query:

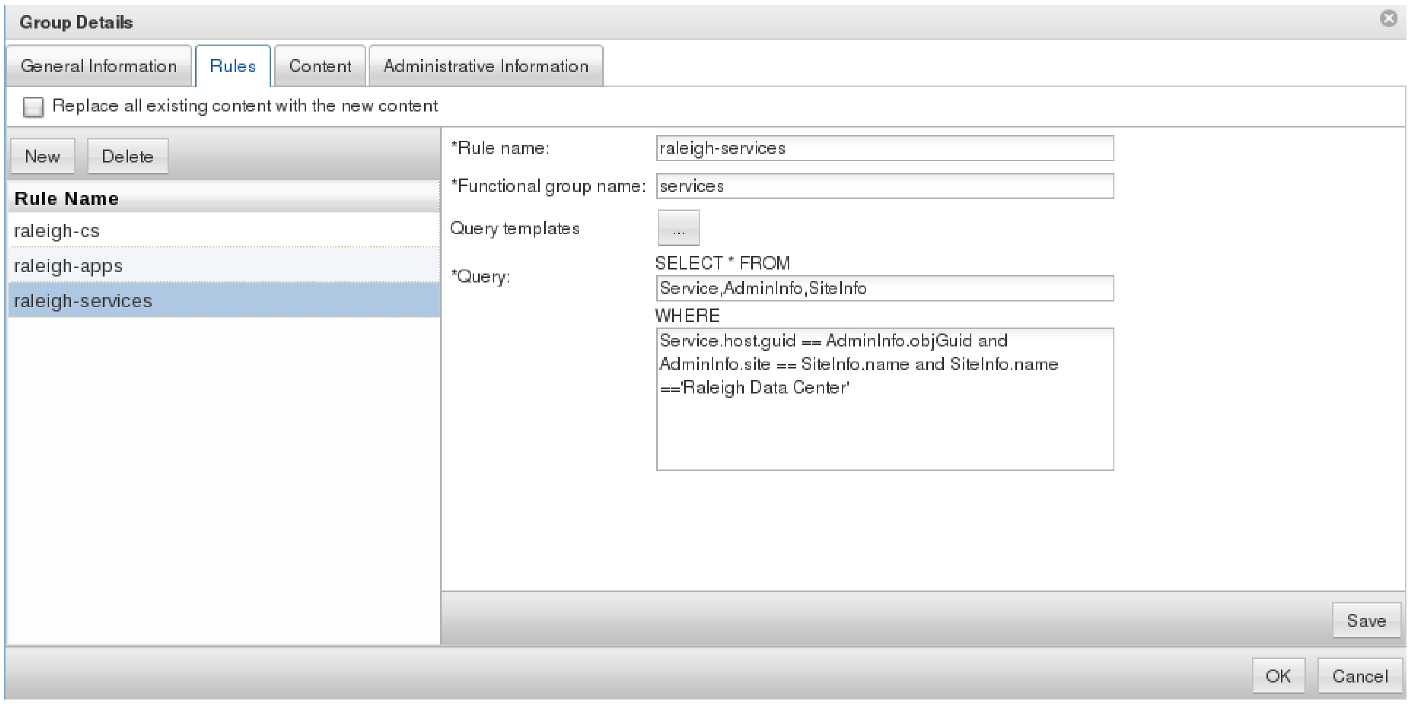
SELECT * FROM AppServer, AdminInfo,SiteInfo

WHERE AppServer.host.guid == AdminInfo.objGuid and AdminInfo.site == SiteInfo.name and SiteInfo.name =='Raleigh Data Center'

Illustration 12: Raleigh Apps Rule

Services

```
SELECT * from Service, AdminInfo, SiteInfo WHERE (Service.host.guid ==
AdminInfo.objGuid and AdminInfo.site == SiteInfo.name and SiteInfo.name
=='Raleigh Data Center' )
```



Replace all existing content with the new content

Rule Name
raleigh-cs
raleigh-apps
raleigh-services

*Rule name: raleigh-services

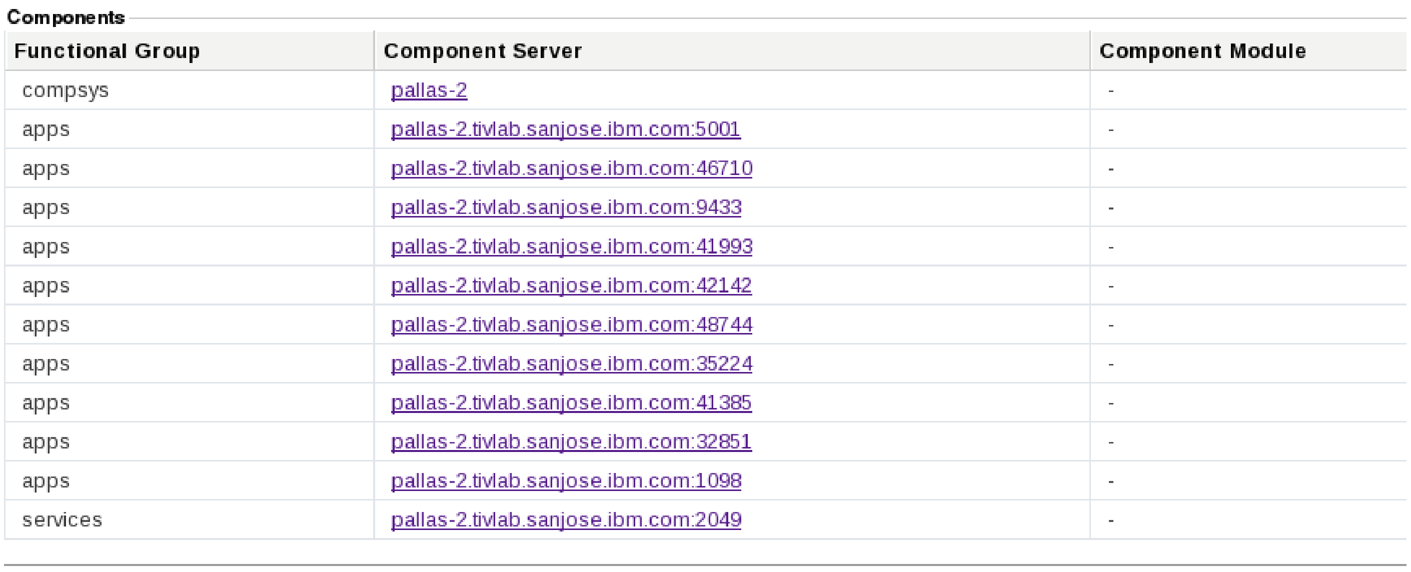
*Functional group name: services

Query templates ...

*Query: SELECT * FROM Service,AdminInfo,SiteInfo WHERE Service.host.guid == AdminInfo.objGuid and AdminInfo.site == SiteInfo.name and SiteInfo.name =='Raleigh Data Center'

Illustration 13: Raleigh Services Rule

The resulting contents of the collection are:



Components

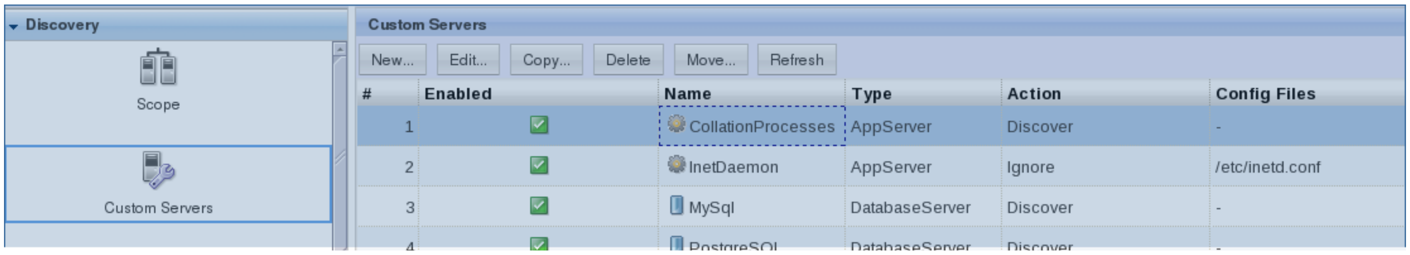
Functional Group	Component Server	Component Module
compsys	pallas-2	-
apps	pallas-2.tivlab.sanjose.ibm.com:5001	-
apps	pallas-2.tivlab.sanjose.ibm.com:46710	-
apps	pallas-2.tivlab.sanjose.ibm.com:9433	-
apps	pallas-2.tivlab.sanjose.ibm.com:41993	-
apps	pallas-2.tivlab.sanjose.ibm.com:42142	-
apps	pallas-2.tivlab.sanjose.ibm.com:48744	-
apps	pallas-2.tivlab.sanjose.ibm.com:35224	-
apps	pallas-2.tivlab.sanjose.ibm.com:41385	-
apps	pallas-2.tivlab.sanjose.ibm.com:32851	-
apps	pallas-2.tivlab.sanjose.ibm.com:1098	-
services	pallas-2.tivlab.sanjose.ibm.com:2049	-

Illustration 14: Raleigh Data Center Contents

6.3 Creating a Business Application for TADDM

Problem: Create a Business Application for my TADDM Streaming Mode infrastructure.

1. Define the Custom Server Templates (TADDM (collation processes) are already defined)



#	Enabled	Name	Type	Action	Config Files
1	✓	CollationProcesses	AppServer	Discover	-
2	✓	InetDaemon	AppServer	Ignore	/etc/inetd.conf
3	✓	MySql	DatabaseServer	Discover	-
4	✓	PostgreSQL	DatabaseServer	Discover	-

Illustration 15: CollationProcesses definition

2. Discover them.

3. Create a group. We are creating a group for the TADDM environment. This will include ComputerSystems where TADDM runs on, its Database and the TADDM Application itself. To do this, we need to add:

ComputerSystems

AppServers

Db2Instance

While not true in every TADDM deployment, this TADDM deployment uses a common name for the ComputerSystem 'pallas' We have a Primary Storage Server on pallas-1 and two discovery servers on pallas-2 and pallas-3 respectively. Our database name is also 'PALLAS'. Since have a common pattern, the rules can be based on the name 'pallas':

ComputerSystem

```
SELECT * FROM ComputerSystem where displayName contains 'pallas'
```

AppServer

```
SELECT * FROM AppServer where host.name contains 'pallas'
```

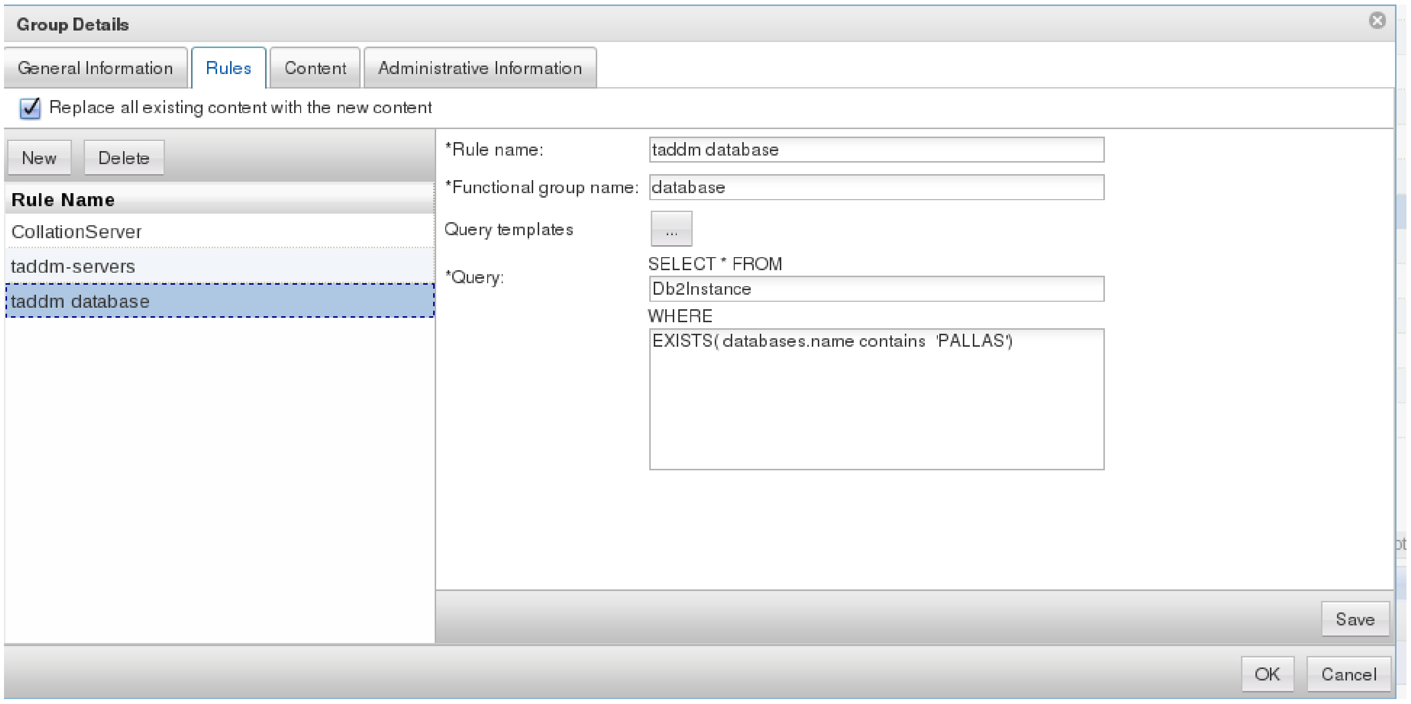
In the above query we are taking advantage of the relationship that an AppServer runs on a host. But we are not limiting the type of AppServer to CollationProcess and are getting all of the AppServers that are running on pallas machines. This could backfire if more AppServers are running on pallas machines.

Database

```
SELECT * FROM Db2Instance where exists (databases.name contains 'PALLAS')
```

To get the database, we need to know the type to select a database based on name because the model for each DB type is slightly different. Knowing that the database is DB2, the above query finds all DB2Instances that have database name 'PALLAS'.

This results in the following rules for the



☑ Replace all existing content with the new content

Rule Name
CollationServer
taddm-servers
taddm database

*Rule name: taddm database
*Functional group name: database
*Query: SELECT * FROM Db2Instance WHERE EXISTS(databases.name contains 'PALLAS')

Illustration 16: TADDM App DB2Instance Definition

The following application is created:

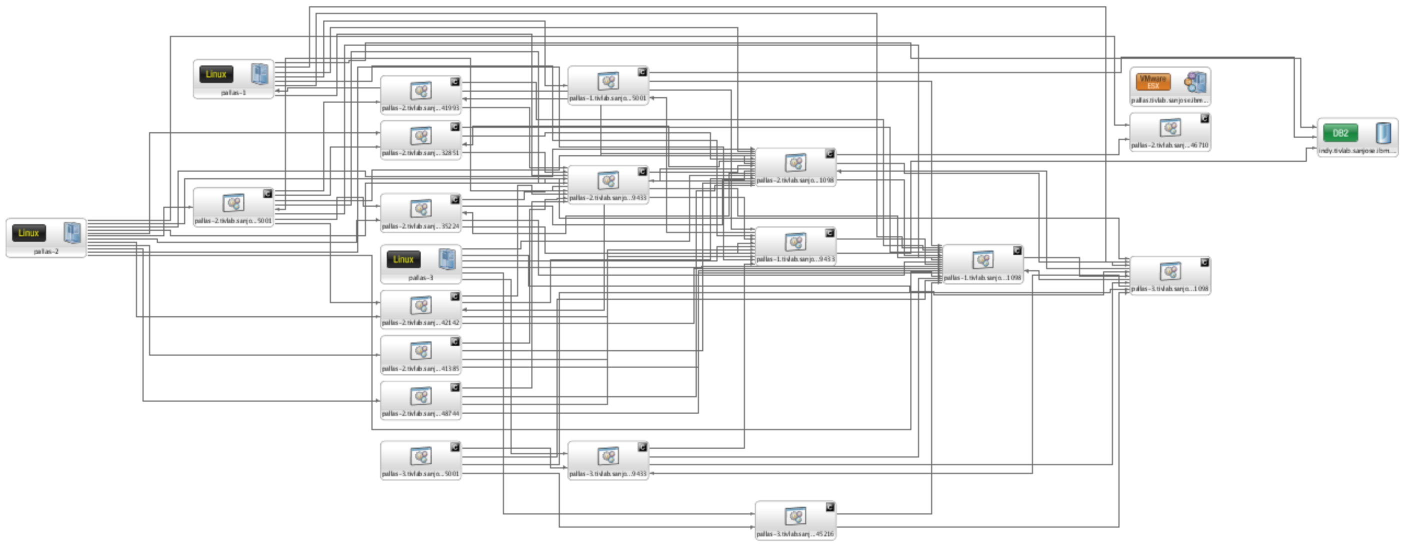

Linux
pallas-1
pallas-2.tivlab.sanj...41993
pallas-1.tivlab.sanjo...5001
pallas-2.tivlab.sanj...32851
VMware ESX
pallas.tivlab.sanjose.ibm...
pallas-2.tivlab.sanj...46710
DB2
indy.tivlab.sanjose.ibm...
pallas-2.tivlab.sanjo...1098
pallas-2.tivlab.sanjo...9433
Linux
pallas-2
pallas-2.tivlab.sanjo...5001
pallas-2.tivlab.sanj...35224
pallas-1.tivlab.sanjo...9433
Linux
pallas-3
pallas-1.tivlab.sanjo...1098
pallas-3.tivlab.sanjo...1098
pallas-2.tivlab.sanj...42142
pallas-2.tivlab.sanj...41385
pallas-2.tivlab.sanj...48744
pallas-3.tivlab.sanjo...5001
pallas-3.tivlab.sanjo...9433
pallas-3.tivlab.sanj...45216

Illustration 17: TADDM Software Topology

Notices