BIZZAPP BEST PRACTICE GUIDE

GUY Thorne

IBM CORPORATION

Table of Contents

Business Applications

A business application is a collection of components that provides business functionality. You can create business applications of individual components, which are related to each other.

For example, Order Management, Inventory Management, and Billing are business applications that might use individual components such as a Java EE application server, LDAP, and a database that runs on the Solaris server.

A business application is a type of custom collection. You can also create the following types of custom collections:

- **Collection**, which is a group of any resources that you can select according to your needs.
- **Access collection**, which is a collection that is used to control the access to configuration items (CIs) and permissions to modify configuration items. You can create access collections only when data-level security is enabled.

The following methods are provided for creating business applications:

- By using grouping patterns in Data Management Portal.
- By using application descriptors.
- By using grouping patterns that are created with Java API and loaded by the bulk load program.

Business applications in TADDM pre-7.3

In the previous TADDM versions, a business application was a flat collection of unconnected **Configuration Items** (CIs). These CIs could be only top-level elements of the Common Data Model. They were grouped into functional groups of elements of the same type. Each CI had to be explicitly added to a business application, either as an instance or by using an MQL rule. It required the user to know exactly which elements formed the business application, and what dependent objects the user should include into the business application.

New approach to generate business applications by using grouping patterns

With TADDM 7.3 or above, the approach to creating business applications changes significantly. A business application is now a graph of connected Core Configuration Items (Core CIs) of types that you specify. The most important elements while building business applications are Core CIs. Core CIs provide the main business value to the specific business application, for example, a Java platform, Enterprise Edition application, or database.

Custom Query

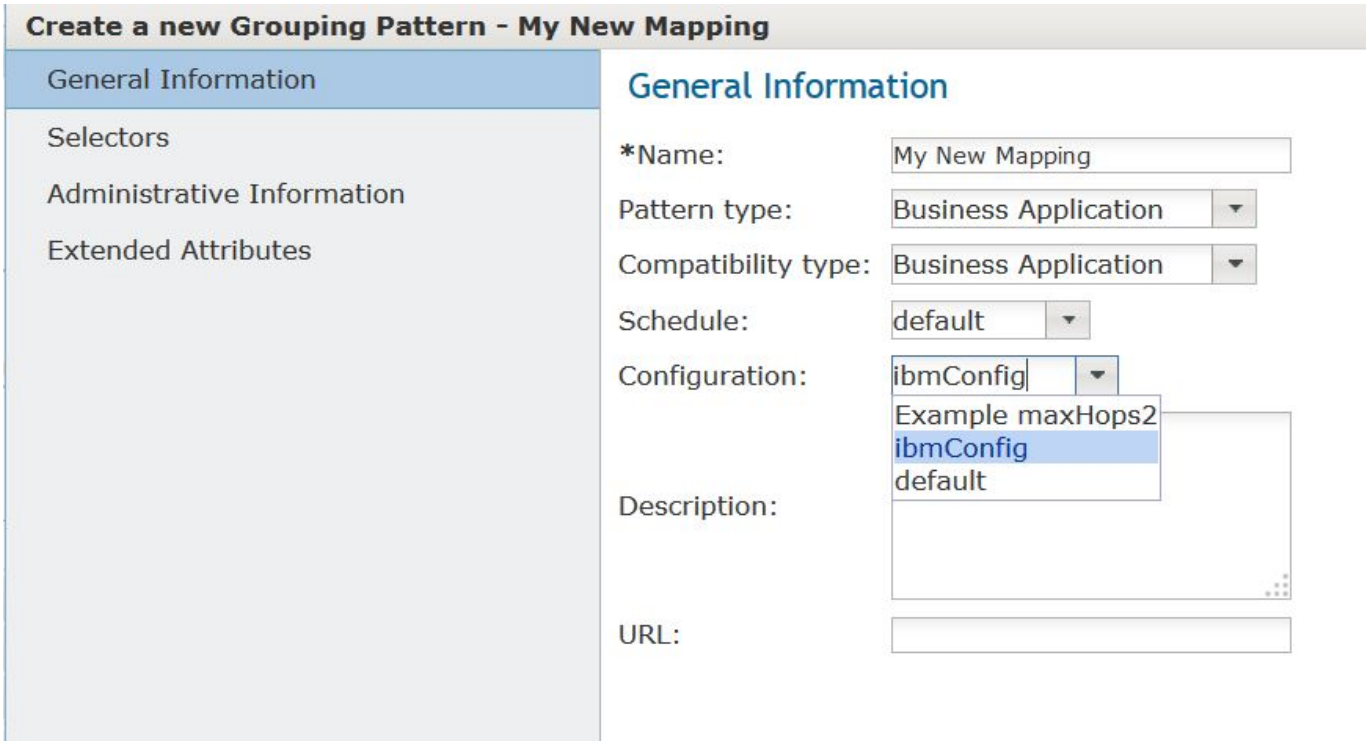
Creating a custom query is an optional initial step in the creation of a business application. Once created, a Custom Query can be used as the Grouping Pattern selector query via the ‘Get from templates’ option. This step is optional as it is also possible to add queries directly in the Grouping Pattern Selectors panel.

Grouping patterns

The power of the new grouping patterns is the data traversal tool that automatically traverses dependencies to add related components to a mapping. However, this power can cause problems by adding too many components. This is controlled through "grouping patterns configurations" that are loaded via the command line **bizappscli** tool. Provided here is a best practices configuration file named **gpConfig.xml** for TADDM 7.3.0.2 that has many important exclusions added to control the data traversal creep that may occur. New sections for the configuration were added in FP1 and FP2. Download the configuration and load it into TADDM using the bizappscli tool. Use myCompanyConfig instead of ibmConfig.

```
$COLLATION_HOME/dist/sdk/bin/bizappscli.sh importConfiguration -c ibmConfig -f
/opt/IBM/taddm/dist/custom/gpConfig.xml
```

Then while using the grouping patterns tool select this new configuration instead of the default.



You may need to customize this configuration to get what you need from your environment. There are comments in the gpConfig.xml file explaining the purpose and result of each change from the default configuration.

See the section in the user's guide for more information about grouping patterns configurations.

Simple Lower-Down Method

This method for data traversal is to keep it simple by only using the Lower-Down template. In this case, you would still need a selector for each type of software component that makes up your application (for example WebSphere or DB2). The Lower-Down template, in conjunction with the grouping pattern configuration above will automatically add the hardware layer and this will allow the proper relationships to display in the topology graph.

Using the simple Lower-Down method should work in most cases. However an alternative example for when Higher Up would be required would be when creating a grouping pattern for a hypervisor. In this case you would use a selector for your hypervisor(s). Then using the Higher Up template would automatically add the computer systems hosted by the hypervisor, as well as all the application servers that are hosted on these computer systems.

Adding computer system components

It is not recommended to add computer system components through a selector, but rather to use the Lower-Down template when adding software components. The exception to this rule is the case where a computer system component does not host any software components. For example, a computer system may be used for storage or may run a batch job that can't be discovered. In this case, you can add a “Computer System” selector to include the hardware layer manually. You may still select the Lower-Down template to include any virtual hardware layer and/or networking layer.

Increase interval for default schedule

Each grouping pattern executes in the background on a default schedule of 4 hours. TADDM runs the selectors and data traversal again to look for any new components and rebuilds the application structure. The default of 4 hours is too frequent, especially for a production environment, and will cause constant thrashing. Discovery does not occur in 4 hour intervals, and business application changes usually don't occur in 4 hour intervals. For this reason it is recommended to increase the interval for the default schedule to at least 24 hours. It should be increased to the maximum amount of time reasonable for your environment.

Here is the command to list the schedules and see the GUID that you will need to update the schedule.

```
$COLLATION_HOME/sdk/bin/bizappscli.sh listSchedules -G
GUID   EXECUTION_GROUP   NAME   INTERVAL   DESCRIPTION
22B7AB1EAAF9399BA75BF4973E160776   default   default   INTERVAL: 4h
```

Here is the command to update the default schedule to 24 hours.

```
$COLLATION_HOME/sdk/bin/bizappscli.sh updateSchedule -g
22B7AB1EAAF9399BA75BF4973E160776 -i 24
```

Selecting Core Configuration Items (CoreCIs)

The Core Configuration Items (CoreCI’s) of a business application should be those items that provide the application functionality. To this end ComputerSystems should generally not be used as a CoreCI.

If discovered, TADDM will automatically find the relationship to the underlying ComputerSystem infrastructure, if a lower down search is used.

Examples of the functional applications that should be used as the CoreCI would be Java Platform, Enterprise Edition application, MQ queues, WebServices, database, etc.

It is also typically best to use more than one CoreCI in your business application. For example, if your application consists of a database, application server and database, you should include all three components in your pattern.

If you are using MQL to search for your CoreCI's you can use the 'api.sh find' option to retrieve the details of like components (for example, 3 db2 servers) and see what attributes make each one unique. See the appendix section for examples. You can use those attributes to compose your MQL for the grouping patterns.

Creating many business applications from one grouping pattern

In previous releases, one application template could produce only one business application. Now you can create not only one business application from one grouping pattern, but you can also create many business applications from one pattern, or a small set of patterns. As a result, it is easy to generalize grouping patterns to produce instances of business applications for multiple environments, for example, the application deployed in production, test, quality assurance, and performance environments. To achieve this, a grouping name expression was created. You can use it to provide a formula to calculate the name of a business application from its core CI. For example, you can use naming conventions to extract specific parts of Cis' names, or any existing attributes that denote the purpose of a specific environment. You can also extend this generalization. For example, you can create a grouping pattern that generates all business applications of a given type, such as Java Platform or Enterprise Edition applications, in all deployment environments.

The grouping name expression resolves to a business application name, which is a unique key that identifies business applications. Therefore, it cannot resolve to a pattern name, as only one business application would be created. To avoid it, you can use the application name as the grouping name expression. In the **Grouping Name Expression** field enter the following expression:

```
${coreCI.displayName}
```

To see the results, click **Test**.

The names are long. To shorten them, instead of the `displayName` attribute, you can use the `name` attribute:

```
${coreCI.name}
```

To see the results, click **Test**.

You can further change some of the names, for example, the ones which have ear file name. You can remove the `.ear` extension by using the following expression:

```
$utils.regex(${coreCI.name}, "(.*)\.ear", "(.*)" )
```

In this expression, the "(.*)\.ear" expression removes the .ear extension from the file name, and the resulting name is the remaining part of that file name. The "(.*)" expression leaves the name as it is. To see the results, click **Test**.

You do not need to modify the names any further. However, there are applications with the same name from different environments, for example the test and production environments. To distinguish them, you can add to their names a prefix with JBoss domain name, or WebSphere Cell name. Use the following expression:

```
$utils.or($coreCI.parent.parent.name,
$coreCI.parent.parent.displayName)$utils.regex(${coreCI.name},
"(.*)\.ear", "(.*)" )
```

WebSphere Cells have always the name attribute set, so the $coreCI.parent.parent.name expression returns all WebSphere Cell names. It is not always the case for JBoss domain, where sometimes the displayName attribute is set. Use the or expression so that the first argument that is not null is returned for each application. To see the results, click **Test**.
You can still have more information, for example server type and a prefix to easily distinguish Java Platform, Enterprise Edition based applications from others. To add the prefix, enter J2EE in front of the expression:

```
J2EE $utils.or($coreCI.parent.parent.name,
$coreCI.parent.parent.displayName)$utils.regex(${coreCI.name},
"(.*)\.ear", "(.*)" )
```

To add the server type, use the following expression:

```
J2EE [${coreCI.parent.productName}]
$utils.or($coreCI.parent.parent.name,
$coreCI.parent.parent.displayName)$utils.regex(${coreCI.name},
"(.*)\.ear", "(.*)" )
```

The server type is taken from the productName attribute of application servers. Because application servers are parents of Java Platform, Enterprise Edition applications, the expression that generates server type is in square brackets. To see the results, click **Test**.
The server type names are too long. You can remove the Application word from the server type name by using the following expression:

```
J2EE [$utils.regex($coreCI.parent.productName, "(.*) Application.*",
"(.*)")] $utils.or($coreCI.parent.parent.name,
$coreCI.parent.parent.displayName)$utils.regex(${coreCI.name},
"(.*)\.ear", "(.*)" )
```

If the name contains the Application word, it is removed by the "(.*) Application.*" expression. If the name does not contain such word, the "(.*)") expression leaves it as it is. To see the results, click **Test**.

If should then be confirmed that “The results look good”. Click **OK**.

For full details about grouping patterns configuration and controlling grouping patterns processing, see https://www.ibm.com/support/knowledgecenter/en/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/UserGuide/t_cmdb_example_scenario2.html?view=embed.

Migration from 7.2.2 and automatic conversion of old business applications

When you upgrade from TADDM 7.2.2 or higher, all business applications are converted automatically into grouping patterns.

The details on how this is done can be found via the following help page:

http://www.ibm.com/support/knowledgecenter/en/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/UserGuide/r_cmdb_busapps_conversion.html

Helpful Troubleshooting MQL Queries

To run any of the following MQL queries you can use:

```
$COLLATION_HOME/sdk/bin/api.sh -u administrator -p collation find "<MQL>"
```

```
where <MQL> contains the query to execute
```

MQL query to list grouping patterns that are using the 'default' grouping patterns configuration.

```
SELECT name FROM GroupingPattern WHERE schedule.name == 'default'
```

MQL query to find active grouping patterns where traversal is enabled and configured to use something other than the simple Lower-Down method described above.

```
SELECT GroupingPattern.name FROM Selector, GroupingPattern
```

```
WHERE Selector.useTraversalTemplate AND ( Selector.goHigherUp OR
Selector.goHigherDown OR Selector.goLowerUp )
```

```
AND NOT Selector.isDisabled AND NOT GroupingPattern.ignore AND Selector.parent.guid
== GroupingPattern.guid
```

MQL query to find grouping patterns that start or end with a blank space. This is helpful for finding copy/paste errors.

```
SELECT name FROM GroupingPattern WHERE name STARTS-WITH ' ' OR name ENDS-WITH ' '
AND NOT ignore
```

MQL query to find grouping patterns where a selector grouping name expression has been changed from the default but does not match the pattern name. This could be a typo or if grouping name expressions are being used then these might be valid.

```
SELECT name FROM GroupingPattern, Selector
```

```
WHERE Selector.groupingNameExpression != '\${patternName}' AND NOT
Selector.isDisabled
```

```
AND Selector.parent.guid == GroupingPattern.guid AND NOT GroupingPattern.ignore
```

```
AND Selector.groupingNameExpression != GroupingPattern.name
```

SQL query to list all business applications with nodes and types.

```
SELECT CSTCOL.GUID_C AS SOURCEGUID, CSTCOL.NAME_C AS NAME, 'Application' AS
SOURCECLASS, 'federates' AS RELATIONSHIP,
```

```
NODE.SOURCEGUID_C AS DESTGUID, NODE.SOURCECOLLATIONTYPE_C AS DESTCLASS FROM
BB_NODE84_V NODE, BB_CUSTOMCOLLECTION23_V CSTCOL
```

```
WHERE NODE.PK__PARENTNODE_C = CSTCOL.PK_C AND CSTCOL.HIERARCHYTYPE_C =
'BusinessApplication'
```

SQL query to find orphaned business application nodes. This occurs when a static selector is used and the underlying component is deleted.

```
SELECT CSTCOL.NAME_C AS NAME, CSTCOL.GUID_C AS SOURCEGUID, 'Application' AS
SOURCECLASS, 'federates' AS RELATIONSHIP,

NODE.SOURCEGUID_C AS DESTGUID, NODE.SOURCECOLLATIONTYPE_C AS DESTCLASS FROM
BB_NODE84_V NODE, BB_CUSTOMCOLLECTION23_V CSTCOL

WHERE NODE.PK__PARENTNODE_C = CSTCOL.PK_C AND CSTCOL.HIERARCHYTYPE_C =
'BusinessApplication'

AND NODE.SOURCEGUID_C NOT IN ( SELECT GUID_X FROM PERSOBJ )
```

BizApp cli tool

To get the full help output execute:

```
$COLLATION_HOME/sdk/bin/bizappscli.sh -u administrator -p collation help
```

Checking you bizapp size

```
$COLLATION_HOME/sdk/bin/bizappscli.sh analytics -r -G -R > routes.out
$COLLATION_HOME/sdk/bin/bizappscli.sh analytics -c -G -R > counts.out
```

The first one will count how many routes there are for each relationship type in your database. If your topology graphs are overcrowded, this is a good method to see at a high level what kind of relationships exist and may be candidates for excluding.

The count option breaks down how many nodes and paths exist per application, which is useful in checking the sizes of individual apps.

Questions and Answers

TADDM 7.3.0 error: "The requested graph has exceeded the number of allowed nodes" error message. How do I fix this?

https://developer.ibm.com/answers/questions/203311/taddm-730-error-the-requested-graph-has-exceeded-t/#answer-203313

What is the use of TADDM 7.3 GroupingNameExpression feature in Grouping Patterns for?

https://developer.ibm.com/answers/questions/319007/whats-is-the-use-of-taddm-73-groupingnameexpressio.html#answer-342837

Application Descriptors question building Business Applications

https://developer.ibm.com/answers/questions/309438/application-descriptors-question-building-business/

I would like to use the TADDM grouping pattern traversal feature in a very limited way, to only include certain classes. Is that possible?

https://developer.ibm.com/answers/questions/295385/i-would-like-to-use-the-taddm-grouping-pattern-tra/

"bizappscli.sh analytics -h" only displays placeholder included in applications, how can I query those not in apps and get the command line?

https://developer.ibm.com/answers/questions/286182/bizappsclish-analytics-h-only-displays-placeholder/

How do I create a customCollection to customCollection relationship in TADDM 7.3?

https://developer.ibm.com/answers/questions/261967/how-do-i-create-a-customcollection-to-customcollec/

Appendix

Common Data Model Documentation

The Common Data Model (CDM) is the definitional language used to integrate understanding and the exchange of data between Tivoli management products concerning resources and components of a customer's business. The CDM is the model used to communicate details about resource instances with the TADDM database.

Documentation of the CDM can be found in both the product install, as well as via Knowledge Center.

Common Data Model documentation installed into the following location:

```
$COLLATION_HOME/sdk/doc/model/CDMWebsite.zip
```

A JavaDocs version is also available in:

```
$COLLATION_HOME/sdk/doc/model/model-javadoc.zip
```

Knowledge Center link for CDM
http://www.ibm.com/support/knowledgecenter/cs/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/SDKDevGuide/c_cmdbsdk_understandingdatamodel.html

Logging

When business applications are generated, log files are also created and stored in the `$COLLATION_HOME/log` directory. If you have problems that are related to business applications, these log files can help you to troubleshoot them.

Logs created when working with grouping patterns and business application in Data Management Portal can be found in:

```
$COLLATION_HOME/log/wlp.log
```

Error messages can be found in:

```
$COLLATION_HOME/log/error.log
```

Logs created when working with business application composition engine and running patterns from the pattern list can be found in:

```
$COLLATION_HOME/log/services/
```

For general logs from scheduling engine, without information from patterns:

```
$COLLATION_HOME/log/services/PatternsSchedulingService.log
```

By default this log will be split, older logs will be saved with .(1-5) extension.

For specific logs for a business application the logs can be found in:

```
$COLLATION_HOME/log/bizapps/<pattern>/<starttime>.log
```

The *<pattern>* folders are created for each grouping pattern in the format *[pattern name-GUID]*, for example `J2EE App pattern-F12AC23451AB3A4FAF58E9187ABF1169`. Inside the pattern folders, you can find log files created for every grouping pattern processing in the format *[time of generation in milliseconds]*`.log`, for example `1416408057548.log`.

Error messages are in the `log/error.log` file and in the same files that contain log messages.

TADDM properties file collation.properties

The collation.properties file contains TADDM configuration items. It can be located:

`$COLLATION_HOME/etc/collation.properties`

The collation.properties file including logging level, topology maxnodes value as well as the jvm arguments.

For bizapp related properties please refer to:

http://www.ibm.com/support/knowledgecenter/en/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/UserGuide/r_cmdb_busapps_properties.html

For general properties please refer to:

http://www.ibm.com/support/knowledgecenter/en/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/AdminGuide/c_cmdb_collationproperties.html

Further customisations to the collation.properties file may exist for each sensor, this will exist within the configuration section of the sensor documentation.

Properties will require a regards apart from the logging level and sensor properties. The latter will be picked up on the next discovery.

Expiring TADDM Data

Via the following Best Practice guide a tool is provided to enable a method to adjust the discovery retention policy. By default TADDM collects and maintains all data that is has discovered indefinitely. The tool provided by this guide allows you to set a retention period for dormant components.

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli+Application+Dependency+Discovery+Manager/page/Expiring+TADDM+Data