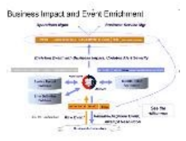


Troubleshooting (tbsm)

Like | Updated yesterday at 12:50 PM by GERALDINE MCCORMACK | Tags: *None* Add tags

Edit Page Actions

Troubleshooting

This section of the wiki includes troubleshooting information for Tivoli Business Service Manager. The following topics are available:

Table of Contents:

APM and XMLToolkit UDF conflict

Both APM and TBSM XMLToolkit use Java UDFs in Db2. The jar files are cataloged at the server level (not the database level). If both TBSM and APM are installing the jars with the same "Db2 name", the last one in wins. The TBSM jar is a subset of the functions defined in the APM jar, therefore we want the APM jar to be in Db2 so that both products work.

If the TBSM Database Configuration Tool is run after APM, then, when finished, you need to run "tbsm_db.sh -s sc -U db2inst1 -f j" from the APM configuration tool to get the jars from APM back into Db2. If you do not do this then you will break APM again because the TBSM tool will have put the TBSM jar back into Db2 and that jar is missing several functions that APM uses.

A fix in APM 8.1.4 IF08 will eliminate the UDF conflict between TBSM and APM, but in the mean time please be cognizant of this. It was fixed under SF ticket TS001850247.

Applet.IsLoadingInstances() function does not exist

In TBSM 6.1 FP1 using Firefox 3.6 an error appears when you launch TBSM from AEL. The error states that applet.isLoadingInstances() does not exist.

Symptom

The symptoms are as follows.

1. An error appears on the screen stating that the applet.isLoadingInstances() function does not exist when the user attempts to launch TBSM from an AEL window.

Cause

Firefox 3.6

Resolving the problem

Pressing the "ok" button will make this issue disappear.

This issue is not present on IE or FireFox 10.

Changing the tbsmadmin password

If for any reason the tbsmadmin password changes (for example, if you switch from file-based user registry to LDAP), you must update the password details for the tbsmadmin user in the RAD_sla.props file. To do so, you must manually encrypt the password and then edit the RAD_sla.props file.

For example, edit the following file:
./JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_sla.props

and add the following lines:

```
# VMM admin user credentials
impact.sla.vmm.admin.username=tbsmadmin
impact.sla.vmm.admin.password={AES}1C09D0508A96864E66E6AE731A68F005
```

This requires a DASH restart.

Console Install inputs

Problem : When installing TBSM Data Server, Dashboard Server or DBConfig using the console install option, the input fields labels are displayed as a combination of numbers and letters rather than a consistent sequential list. This is a known issue with the underlying code.

Solution : When selecting inputs, first enter the number or letter corresponding to the value you want to select. When prompted, enter the value you wish to input and hit enter.
The newly input value should be displayed in the upper list, next to its corresponding label.
Ensure all values are correct before using N key to move on to the next page of the installer.

Event Integration Facility probe installation prompts

The Tivoli EIF Probe is used to support event flows from event source products such as Tivoli Enterprise Console, IBM Tivoli Monitoring , and NetView.

Although you can install the Tivoli EIF probe on the TBSM 4.1.1 server, it is strongly recommended that the probe be installed at the event source server. One advantage to installing the probe on the event source server is the probe's capability to provide a secure sockets layer (SSL) connection between itself and the TBSM 4.1.1 server. The details of the SSL connection are described in the Netcool/OMNIbus 7.1 Installation Guide.

Starting the EIF probe installer

You can use the TBSM 4.1.1 Launchpad to install the Tivoli EIF probe, or you can install the probe from the TBSM 4.1.1 product DVD.

From the launch pad, Select **Install Tivoli EIF Probe** and click **Run the Tivoli EIF Probe installation program**.

To start from the DVD on a UNIX® system:

1. Change to the platform/TBSM directory, where platform is the operating system (such as Linux®, AIX®, HPux, or Solaris™).
2. Type ./tbsmEifProbe.

To start from the DVD on a Windows system, complete the following steps:

1. Change to the Win32\TBSM directory.
2. Type tbsmEifProbe.exe.

ObjectServer install prompts

Note: For localhost to work correctly in TBSM 4.1.1 on UNIX hosts, the /etc/hosts file needs to be configured so that when you ping localhost, you get the computer's IP address, not 127.0.0.1.

The **Configured Value** column is provided so that you can add your configured values. To create a PDF of this page, click here and follow the instructions.

Prompt	Description	Default	Configured Value
Name	ObjectServer name. Required.	NCOMS	
Hostname	Required		
Port	Must be in the range 1241-65535. Required.	4100	

Probe listener install prompts

Prompt	Description	Default	Configured Value
Probe Listening port	This is the port number where an event source product such as TEC, ITM, and NetView have been configured to forward events to. Must be in the range 1241 - 65535. Required.	5530	
Setup Tivoli EIF probe for failover	Select this option to have the probe's property file setup for failover. If you select this option, you must setup the primary ObjectServer for failover.	Disabled	
Setup Tivoli EIF probe as a Windows service.	Optional and only available on Windows installations. Select this option to have the server create a Windows service definition that will start the probe when the Windows OS is started. The name of the service is NCO NONNATIVE Probe (NCOTivoliEIFProbe).	Disabled	

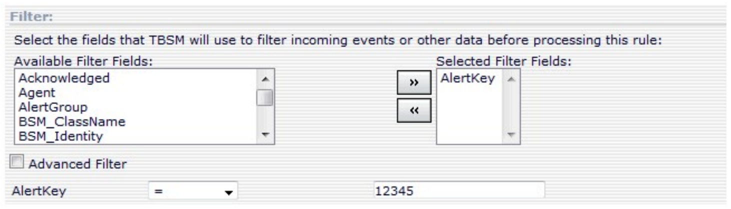
Event Not Affecting Service in TBSM

I had a bit of trouble with this event problem and then I thought I had it reproduced it in the lab, but it turned out I had the wrong setting for my rule.

For starters none of my events would match... and I discovered that the RAD_FilterIDList was set to an old value in the Omnibus event, so TBSM never bothered to try and find my new rules. When I cleared this field I could see this in my trace.log:

getRawAttributeWrappersByEvaluatingFilters No Filters found in Event Container so must get filters.

I have 2 Incoming Status Rules, both with the same filter. AlertKey = '12345'. One Rule sets a text output based on the Summary Field and the other sets output based on BSM_Identity.


Filter:
Select the fields that TBSM will use to filter incoming events or other data before processing this rule:
Available Filter Fields: Acknowledged, Agent, AlertGroup, BSM_ClassName, BSM_Identity
Selected Filter Fields: AlertKey

AlertKey = 12345

In the logs, I see TBSM matching to my event filter:

```
MemoryBasedMatchingRawAttributeGetter          2 testThresholdMatch Testing AlertKey with comparator = '12345'.
MemoryBasedMatchingRawAttributeGetter                    2 testThresholdMatch fieldMatched is: true.
```

Next it gets the output expression field from the Event. My output expression for one rule is set to Summary field, which has a value of zzzzzzz. My output expression for the other rule is set to BSM_Identity field, which has a value of mmmmmmm.

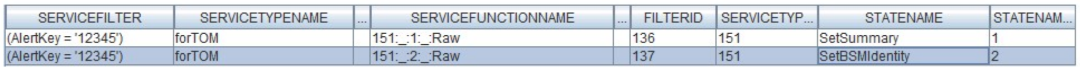
```
MemoryBasedMatchingRawAttributeGetter 2 getThresholdValue EXIT ^tag2^S^mmmmmmm
```

Next it sets the RAD_FilterIDList for the event to be that of the rule.

```
addFilterIDToEvent New RAD_FilterIDList for Event is to set to: 137.
addFilterIDToEvent New RAD_FilterIDList for Event is to set to: 137,136.
```

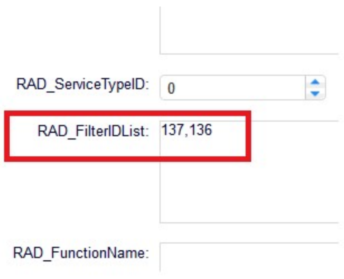
Note: I have 2 rules (as per customer) and each will have their own Filter ID. Both should are added to the RAD_FilterIDList. To see which filter IDs related to which rules, use the SQL below (replace forTOM with your own template name):

```
    select filterid , servicefilter from tbsmbase.eventmappingfilter where servicetypename = 'forTOM'
```



SERVICEFILTER	SERVICETYPENAME	...	SERVICEFUNCTIONNAME	...	FILTERID	SERVICETYP...	STATENAME	STATENAM...
(AlertKey = '12345')	forTOM		151:_:1:_:Raw		136	151	SetSummary	1
(AlertKey = '12345')	forTOM		151:_:2:_:Raw		137	151	SetBSMIdentity	2

You can see this in the event.



RAD_ServiceTypeID: 0

RAD_FilterIDList: 137,136

RAD_FunctionName:

But only one is firing. Only 151:_:1:_:Raw fires. I have one SLAM event: This is the one for filter ID 136. For rule setSummary.

```
   Event based attribute SetSummary of template forTOM and service aaaaaaaaaaaaaaa has value zzzzzzz
```

Note: SetSummary is the first rule. It has attributeID 1 and a filter = 136.
   SetBSMIdentity is the second rule. It has attributeID 2 and a filter = 137.

TBSM finds the correct service (service ID 338) for both rules.

```
getCommaSeparatedInstanceIDs TagId is 151. attributeID is 2. eventMatchingString is aaaaaaaaaaaaaaa. List of Service Instance Ids which match raw attribute is: 338.
getCommaSeparatedInstanceIDs TagId is 151. attributeID is 1. eventMatchingString is aaaaaaaaaaaaaaa. List of Service Instance Ids which match raw attribute is: 338.
```

Both are sent to the state model - note the BSM_Identity was sent first but this one gives an error because I had set the rule to be Numeric Rule so is did not work!

```
sendInstancesToStateModel1 Status/Arguments are mmmmmmm.
sendInstancesToStateModel1 NOTE ^Send to statemodel aaaaaaaaaaaaaaa^n^
[2/22/19 6:09:18:139 PST] 00000c91 id=         com.micromuse.sla.impact.RADEventProcessor                1 sendInstancesToStateModel1 ENTER^^T^For input string: "mmmmmmm"
java.lang.NumberFormatException: For input string: "mmmmmmm"
```

```
sendInstancesToStateModel1 Status/Arguments are zzzzzzz.
sendInstancesToStateModel1 NOTE ^Send to statemodel aaaaaaaaaaaaaaa^n^
passToStateModel ENTER^^o^{{"Prepared by TBSMOMNIbusEventReader"=(EventContainer=(RAD_TimeWindo...
passToStateModel Status is zzzzzzz and text based.
setVarsInContexts eventInstanceID is: 151:_:1:_:Raw++**NCOMS:_:9246.
setTextAttributeValue ENTER^^o^{{151:_:1:_:Raw,151:_:1:_:Raw++**NCOMS:_:9246,zzzzzzz,1550
updateServiceDeps About to update service_deps in memory and at object server for SIB: aaaaaaaaaaaaaaa.
```

So next I deleted the bad SetBSMIdentity Rule and created a new one, Then I updated the Event to clear the RAD_FilterIDList , and change the Severity and SeenByTBSM = 0.

SetBSMIdentity2 is the second rule. It has attributeID 3 and a filter = 138

This time is worked beautifully (ref g3) - I have 2 SLAM events for both rules.

```
   Event based attribute SetBSMIdentity2 of template forTOM and service aaaaaaaaaaaaaaa has value ggggggggg
   Event based attribute SetSummary of template forTOM and service aaaaaaaaaaaaaaa has value ffffffffffffffff
```

And if I update the event by updating the SeenByTBSM, and Severity Fields (ref g4) : they are both updated.

```
   Event based attribute SetBSMIdentity2 of template forTOM and service aaaaaaaaaaaaaaa has value gggggggggUPDATED
   Event based attribute SetSummary of template forTOM and service aaaaaaaaaaaaaaa has value ffffffffffffffffUPDATED
```

So now that I know it is working on mine, I looked at the trace log from the customer. I could not find Serial 12612621 but I did find 12612609.

Note: the RAD_FilterIDList is already set. To see which template rules these belong to use this SQL : =

```
   select * from tbsmbase.eventmappingfilter where filterid in (152,37,162,212,149,213,41,153)
```

Template: OPS-Reported-State-T3 must have ID=165 (template is the same as tag) because the only 2 rules/attriubte which match are for this one. The service instance id is 408.

```
[2/21/19 23:47:49:290 GMT] 0000025e id=00000000 GlobalInstanceFieldStore                              2 getCommaSeparatedInstanceIDs TagId is 165. attributeID is 1. eventMatchingString is G:S:WO:NA:EM Event Management. List of Service Instance Ids which match raw attribute is: 408.
[2/21/19 23:47:49:305 GMT] 0000025e id=00000000 GlobalInstanceFieldStore                              2 getCommaSeparatedInstanceIDs TagId is 165. attributeID is 2. eventMatchingString is G:S:WO:NA:EM Event Management. List of Service Instance Ids which match raw attribute is: 408.
```

However, I see an issue when sending one of the values - and the issue seems to be related to setting the status of the Parent service.

Below is the error causing the issue:

```
sendInstancesToStateModel1 Status/Arguments are Inactive.
passToStateModel Status is Inactive and text based.
setVarsInContexts eventInstanceID is: 165:_:1:_:Raw++**GOESR_P:_:12612609.
setStatus Setting internal status lastValue is: Active (may be null if never set). For attribute: 165:_:1:_:Raw.
stateChanged About to update parents for current SIB.
sendInstancesToStateModel1 ENTER^^T^For input  string: "null"
java.lang.NumberFormatException: For input string: "null"
       at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2056)
       at sun.misc.FloatingDecimal.parseDouble(FloatingDecimal.java:123)
       at java.lang.Double.parseDouble(Double.java:549)
```

Below shows a successful update:

```
sendInstancesToStateModel1 Status/Arguments are BOB.
passToStateModel Status is BOB and text based.
setVarsInContexts eventInstanceID is: 165:_:2:_:Raw++**GOESR_P:_:12612609.
setStatus Setting internal status lastValue is: BOB (may be null if never set). For attribute: 165:_:2:_:Raw.
stateChanged About to update parents for current SIB.
setStatus Setting internal status lastValue is: 0 (may be null if never set). For attribute: 165:_:0:_:OverallAttribute.
```

The update is failing for one rule because it is trying to set the status on the parent. There must be some difference between the rules, i.e. the failing one is referenced from a dependancy rule on another template. The child service name is "G:S:WO:NA:EM Event Management". If you can find the parent of this and look at the dependancy rules for its templates it might give the answer. i.e. if the child rule does not set Status, only a text output, then the parent dependancy rule cannot be expecting double/numeric. Or, easier still, get the radshell export and we can recreate in the lab.

FileNotFoundException - ja_JP.js

Problem:
After installing TBSM6.2, the customer opened IBM Dashboard Application Services Hub page.
[Service Administration] page -> [TBSM Charting] portlet shows the following error.

```
Error collecting data visualization input data.
Failed creating dataset.
Unable to load /ibm/tivoli/rest/providers/Impact_TBSMCLUSTER.provider/datasources/IMPACT_POLICY_TBSMEventSummaryData/datasets/TBSMEventSummaryData?d=1552982108868&r=0.5348481921725618&optimize=true
param_executePolicy=true&param_ServiceInstanceName=default&param_pageId=1552982108868&param_refId=0.4375004412125408&param_tz=Asia%2FTokyo status: 0
```

Error in SystemOut.log in TBSM_dash:

```
E com.ibm.ws.webcontainer.filter.FilterInstanceWrapper doFilter SRVE8109W: ????? tbsmStaticFileFilter-ui ???????????????????????: java.io.FileNotFoundException: SRVE0190E: ???? /netcool/scripts/language/ja_JP.js
?????????
```

Solution:

As a workaround, go to the following directory as the user installing TBSM:

1) cd /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/netcool/scripts/language/

2) cp ja.js ja_JP.js

Then backup the following file:

/opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/WEB-INF/portlet.xml

And then add the following lines to the file:

```
<supported-locale>ja_JP</supported-locale>
```

Please restart TBSM

IBM Support Assistant 4.0.2 Add-on for TBSM 4.2

This presentation describes the TBSM 4.2 specific support provided with the IBM Support Assistant version V4 add-on.

Found here

Impact Data Server is going down with decryption errors in impactserver.log

Problem: Impact Data Server is going down with decryption errors in impactserver.log

Solution: Follow the steps in the following tech note under the section "Multi-threading issue when decrypting passwords"

http://ibm.biz/Networkandconnectivityissues

java.lang.NullPointerException in messages.log

Problem:
Below error is generated in TBSM Data Server log file:
java.lang.NullPointerException in messages.log

The stack trace may show the below:

```
[18/06/18 09:39:27:796 EDT] 0000002a SystemErr                       R java.lang.NullPointerException
[18/06/18 09:39:27:797 EDT] 0000002a SystemErr                       R    at com.micromuse.sla.configserver.BeanContextManager.<init>(BeanContextManager.java:78)
[18/06/18 09:39:27:798 EDT] 0000002a SystemErr                       R    at com.micromuse.sla.configserver.BeanContextManager.getInstance(BeanContextManager.java:59)
[18/06/18 09:39:27:798 EDT] 0000002a SystemErr                       R    at com.micromuse.sla.configserver.RADServerFacadeImpl.<init>(RADServerFacadeImpl.java:52)
[18/06/18 09:39:27:799 EDT] 0000002a SystemErr                       R    at java.lang.J9VMInternals.newInstanceImpl(Native Method)
[18/06/18 09:39:27:805 EDT] 0000002a SystemErr                       R    at java.lang.Class.newInstance(Class.java:1781)
[18/06/18 09:39:27:806 EDT] 0000002a SystemErr                       R    at com.ibm.ws.jaxws.support.JaxWsInstanceManager.createInstance(JaxWsInstanceManager.java:64)
[18/06/18 09:39:27:806 EDT] 0000002a SystemErr                       R    at com.ibm.ws.jaxws.support.JaxWsInstanceManager.createInstance(JaxWsInstanceManager.java:43)
[18/06/18 09:39:27:822 EDT] 0000002a SystemErr                       R    at com.ibm.ws.jaxws.web.POJOJaxWsWebEndpoint.init(POJOJaxWsWebEndpoint.java:79)
[18/06/18 09:39:27:823 EDT] 0000002a SystemErr                       R    at com.ibm.ws.jaxws.webcontainer.LibertyJaxWsServlet.init(LibertyJaxWsServlet.java:58)
[18/06/18 09:39:27:823 EDT] 0000002a SystemErr                       R    at com.ibm.ws.webcontainer.servlet.ServletWrapper.init(ServletWrapper.java:297)
[18/06/18 09:39:27:823 EDT] 0000002a SystemErr                       R    at com.ibm.ws.webcontainer.servlet.ServletWrapper.loadOnStartupCheck(ServletWrapper.java:1393)
[18/06/18 09:39:27:824 EDT] 0000002a SystemErr                       R    at com.ibm.ws.webcontainer.webapp.WebApp.doLoadOnStartupActions(WebApp.java:1153)
[18/06/18 09:39:27:824 EDT] 0000002a SystemErr                       R    at com.ibm.ws.webcontainer.webapp.WebApp.commonInitializationFinally(WebApp.java:1121)
[18/06/18 09:39:27:828 EDT] 0000002a SystemErr                       R    at com.ibm.ws.webcontainer.webapp.WebApp.initialize(WebApp.java:1022)
[18/06/18 09:39:27:828 EDT] 0000002a SystemErr                       R    at com.ibm.ws.webcontainer.webapp.WebApp.initialize(WebApp.java:6550)
```

Solution:
There is no known functional issue with this exception.

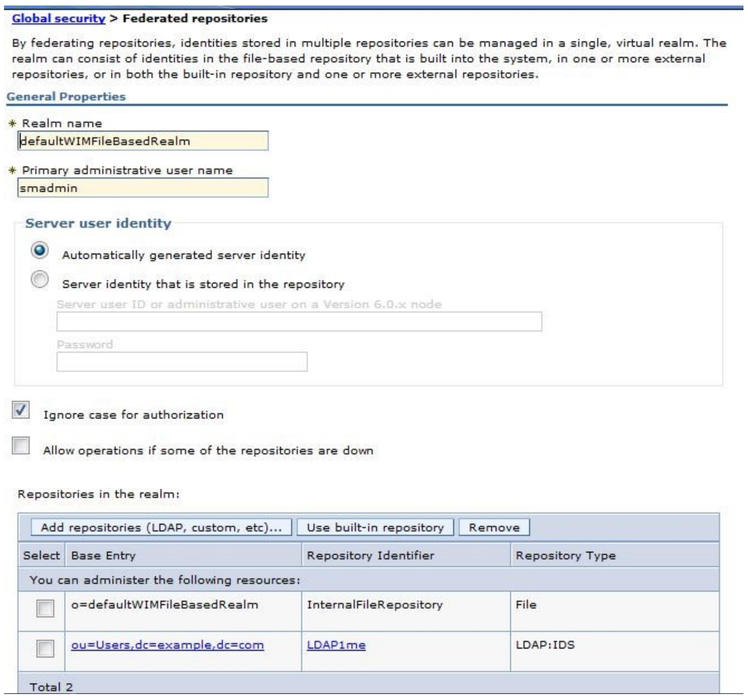
launchpad install fails for root user

Installing TBSM as root user on Unix is valid, however with root user, only console mode or silent install is supported on the following Operating Systems:

v AIX 6.1.4 with SP5, or later for PA-RISC platforms, 32-bit and 64-bit
v AIX 7.1 POWER® System 64-bit
v SLES 10 Intel x86, 32-bit and 64-bit
v SLES 11 Intel x86, 32-bit and 64-bit
v Solaris 10 for Sun Sparc platforms, 32-bit and 64-bit

The launchpad install will not succeed on the above operating systems, if launch by the root user.

LDAP User setup for TBSM



Global security > Federated repositories

By federating repositories, identities stored in multiple repositories can be managed in a single, virtual realm. The realm can consist of identities in the file-based repository that is built into the system, in one or more external repositories, or in both the built-in repository and one or more external repositories.

General Properties

* Realm name: defaultWIMFileBasedRealm

* Primary administrative user name: smadmin

Server user identity

- [✓] Automatically generated server identity
- [ ] Server identity that is stored in the repository
  - Server user ID or administrative user on a Version 6.0.x node
  - Password

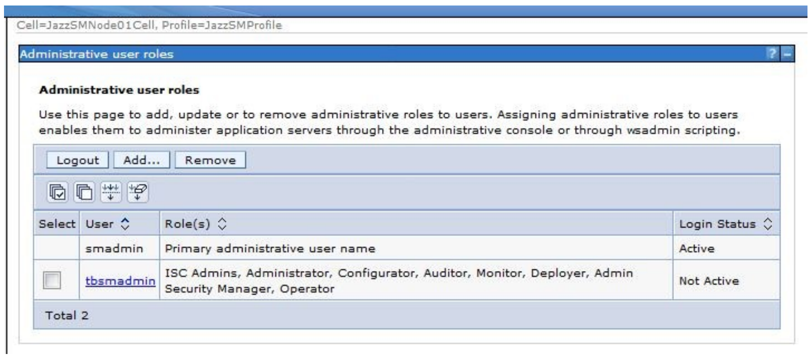
[✓] Ignore case for authorization

[ ] Allow operations if some of the repositories are down

Repositories in the realm:

Add repositories (LDAP, custom, etc)... | Use built-in repository | Remove

Select	Base Entry	Repository Identifier	Repository Type
You can administer the following resources:			
☐	o=defaultWIMFileBasedRealm	InternalFileRepository	File
☐	ou=Users,dc=example,dc=com	LDAP1me	LDAP:IDS
Total 2			



Cell=JazzSMNode01Cell, Profile=JazzSMProfile

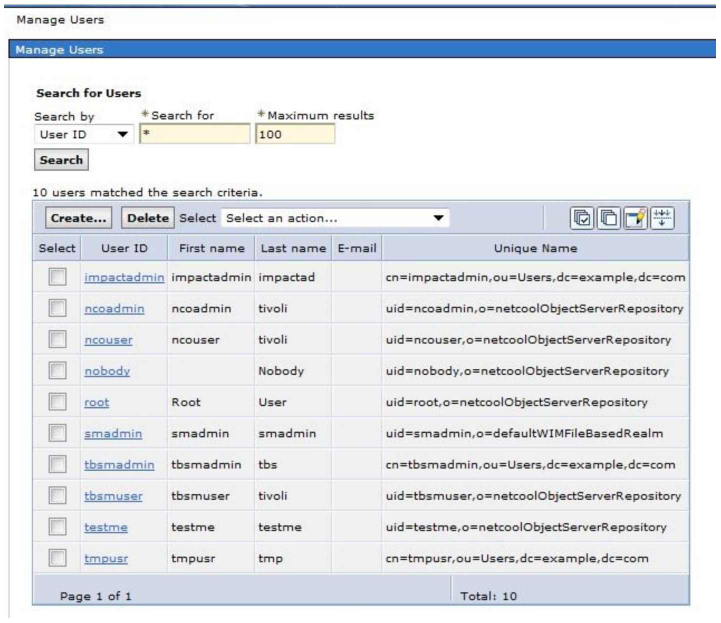
Administrative user roles

Administrative user roles

Use this page to add, update or to remove administrative roles to users. Assigning administrative roles to users enables them to administer application servers through the administrative console or through wsadmin scripting.

Logout | Add... | Remove

Select	User	Role(s)	Login Status
	smadmin	Primary administrative user name	Active
☐	tbsmadmin	ISC Admins, Administrator, Configurator, Auditor, Monitor, Deployer, Admin Security Manager, Operator	Not Active
Total 2			



Manage Users

Manage Users

Search for Users

Search by: User ID | * Search for: * | * Maximum results: 100

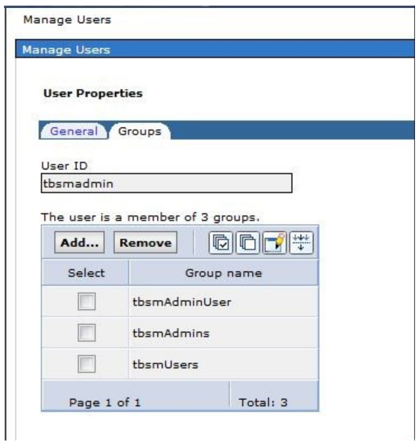
Search

10 users matched the search criteria.

Create... | Delete | Select: Select an action...

Select	User ID	First name	Last name	E-mail	Unique Name
☐	impactadmin	impactadmin	impactad		cn=impactadmin,ou=Users,dc=example,dc=com
☐	ncoadmin	ncoadmin	tivoli		uid=ncoadmin,o=netcoolObjectServerRepository
☐	ncouser	ncouser	tivoli		uid=ncouser,o=netcoolObjectServerRepository
☐	nobody		Nobody		uid=nobody,o=netcoolObjectServerRepository
☐	root	Root	User		uid=root,o=netcoolObjectServerRepository
☐	smadmin	smadmin	smadmin		uid=smadmin,o=defaultWIMFileBasedRealm
☐	tbsmadmin	tbsmadmin	tbs		cn=tbsmadmin,ou=Users,dc=example,dc=com
☐	tbsmuser	tbsmuser	tivoli		uid=tbsmuser,o=netcoolObjectServerRepository
☐	testme	testme	testme		uid=testme,o=netcoolObjectServerRepository
☐	tmpusr	tmpusr	tmp		cn=tmpusr,ou=Users,dc=example,dc=com

Page 1 of 1 | Total: 10



Manage Users

Manage Users

User Properties

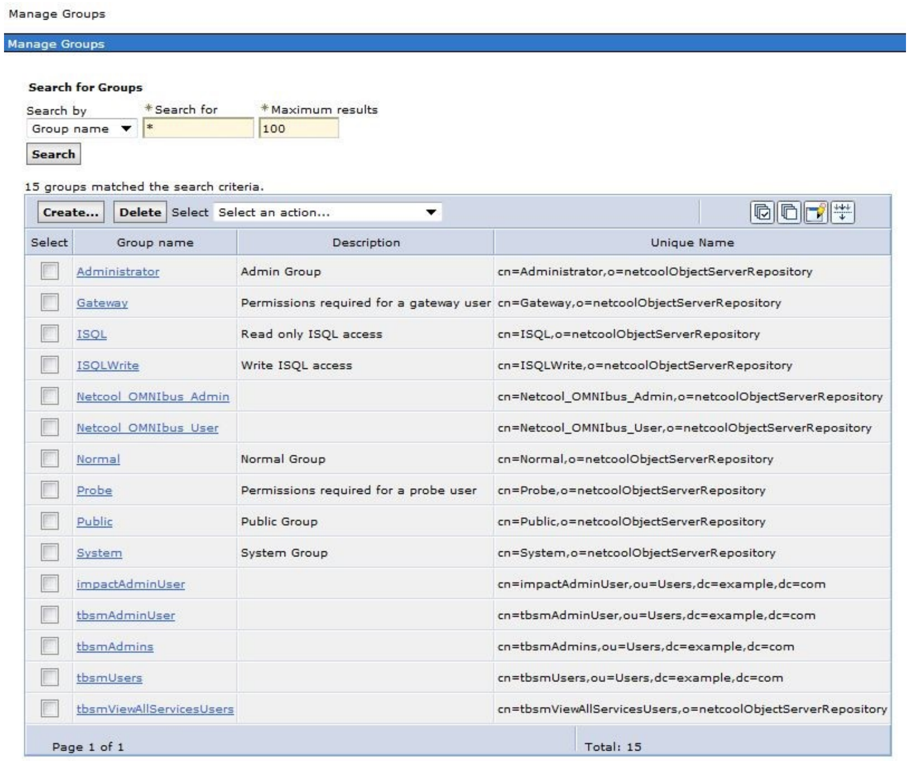
General | Groups

User ID: tbsmadmin

The user is a member of 3 groups.

Select	Group name
☐	tbsmAdminUser
☐	tbsmAdmins
☐	tbsmUsers

Page 1 of 1 Total: 3



Manage Groups

Manage Groups

Search for Groups

Search by: Group name | * Search for: * | * Maximum results: 100

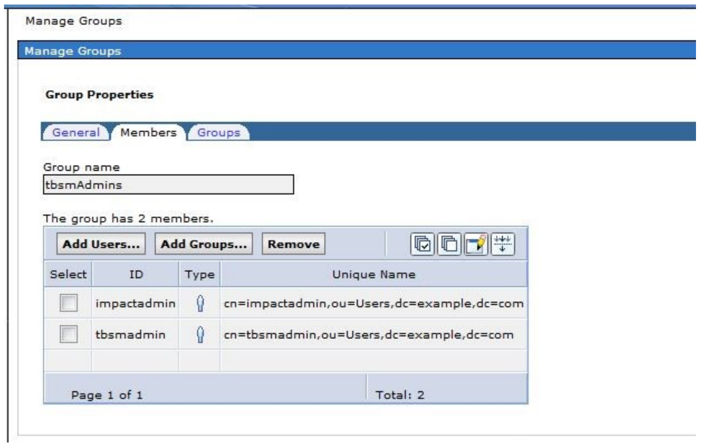
Search

15 groups matched the search criteria.

Create... Delete Select Select an action...

Select	Group name	Description	Unique Name
☐	Administrator	Admin Group	cn=Administrator,o=netcoolObjectServerRepository
☐	Gateway	Permissions required for a gateway user	cn=Gateway,o=netcoolObjectServerRepository
☐	ISQL	Read only ISQL access	cn=ISQL,o=netcoolObjectServerRepository
☐	ISQLWrite	Write ISQL access	cn=ISQLWrite,o=netcoolObjectServerRepository
☐	Netcool_OMNIbus_Admin		cn=Netcool_OMNIbus_Admin,o=netcoolObjectServerRepository
☐	Netcool_OMNIbus_User		cn=Netcool_OMNIbus_User,o=netcoolObjectServerRepository
☐	Normal	Normal Group	cn=Normal,o=netcoolObjectServerRepository
☐	Probe	Permissions required for a probe user	cn=Probe,o=netcoolObjectServerRepository
☐	Public	Public Group	cn=Public,o=netcoolObjectServerRepository
☐	System	System Group	cn=System,o=netcoolObjectServerRepository
☐	impactAdminUser		cn=impactAdminUser,ou=Users,dc=example,dc=com
☐	tbsmAdminUser		cn=tbsmAdminUser,ou=Users,dc=example,dc=com
☐	tbsmAdmins		cn=tbsmAdmins,ou=Users,dc=example,dc=com
☐	tbsmUsers		cn=tbsmUsers,ou=Users,dc=example,dc=com
☐	tbsmViewAllServicesUsers		cn=tbsmViewAllServicesUsers,o=netcoolObjectServerRepository

Page 1 of 1 Total: 15



Manage Groups

Manage Groups

Group Properties

General | Members | Groups

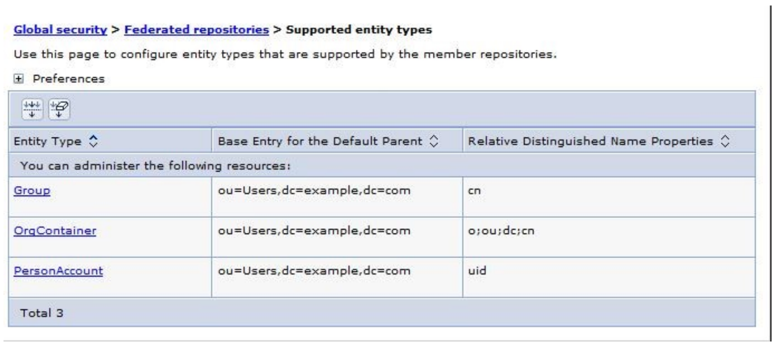
Group name: tbsmAdmins

The group has 2 members.

Add Users... Add Groups... Remove

Select	ID	Type	Unique Name
☐	impactadmin	👤	cn=impactadmin,ou=Users,dc=example,dc=com
☐	tbsmadmin	👤	cn=tbsmadmin,ou=Users,dc=example,dc=com

Page 1 of 1 Total: 2

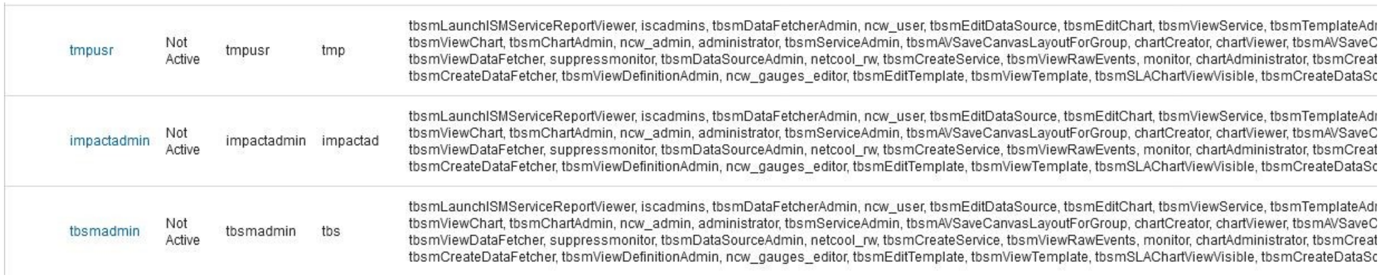


Global security > Federated repositories > Supported entity types

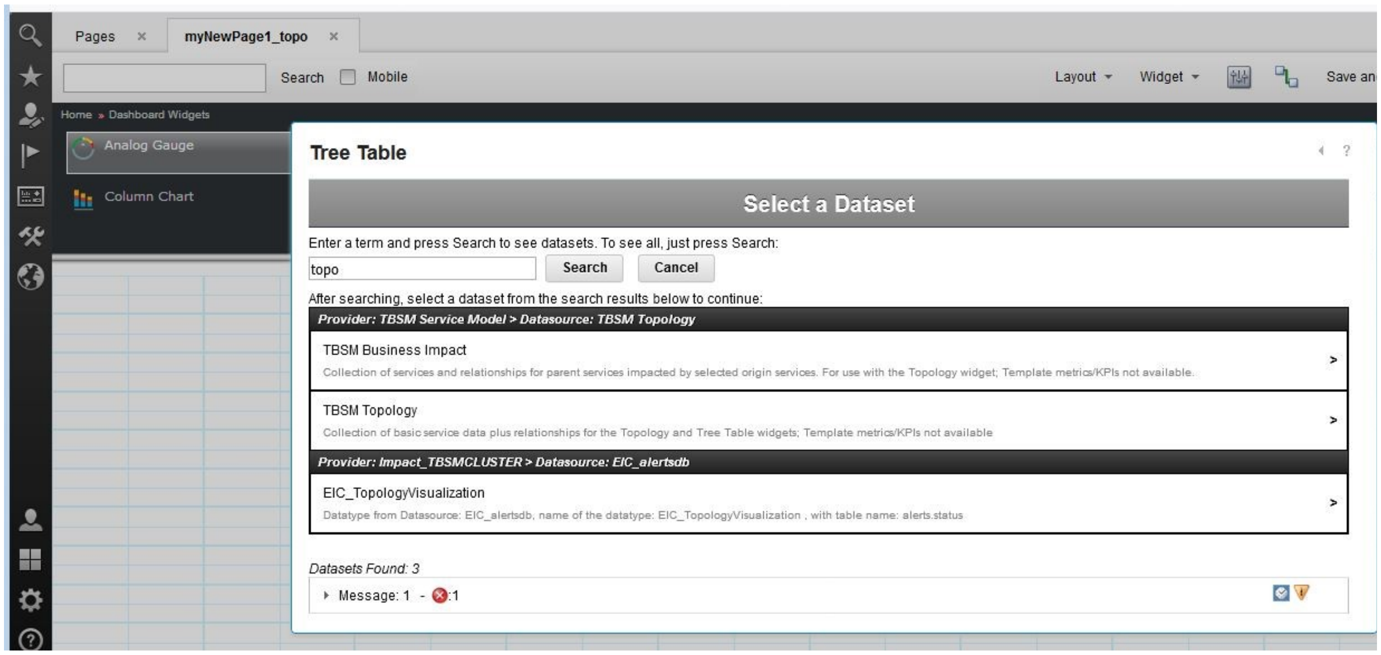
Use this page to configure entity types that are supported by the member repositories.

⊞ Preferences

Entity Type	Base Entry for the Default Parent	Relative Distinguished Name Properties
You can administer the following resources:		
Group	ou=Users,dc=example,dc=com	cn
OrgContainer	ou=Users,dc=example,dc=com	o;ou;dc;cn
PersonAccount	ou=Users,dc=example,dc=com	uid
Total 3		



tmpusr	Not Active	tmpusr	tmp	tbsmLaunchISMServiceReportViewer, iscadmins, tbsmDataFetcherAdmin, ncw_user, tbsmEditDataSource, tbsmEditChart, tbsmViewService, tbsmTemplateAdı tbsmViewChart, tbsmChartAdmin, ncw_admin, administrator, tbsmServiceAdmin, tbsmAVSaveCanvasLayoutForGroup, chartCreator, chartViewer, tbsmAVSaveC tbsmViewDataFetcher, suppressmonitor, tbsmDataSourceAdmin, netcool_rw, tbsmCreateService, tbsmViewRawEvents, monitor, chartAdministrator, tbsmCreat tbsmCreateDataFetcher, tbsmViewDefinitionAdmin, ncw_gauges_editor, tbsmEditTemplate, tbsmViewTemplate, tbsmSLAChartViewVisible, tbsmCreateDataSc
impactadmin	Not Active	impactadmin	impactad	tbsmLaunchISMServiceReportViewer, iscadmins, tbsmDataFetcherAdmin, ncw_user, tbsmEditDataSource, tbsmEditChart, tbsmViewService, tbsmTemplateAdı tbsmViewChart, tbsmChartAdmin, ncw_admin, administrator, tbsmServiceAdmin, tbsmAVSaveCanvasLayoutForGroup, chartCreator, chartViewer, tbsmAVSaveC tbsmViewDataFetcher, suppressmonitor, tbsmDataSourceAdmin, netcool_rw, tbsmCreateService, tbsmViewRawEvents, monitor, chartAdministrator, tbsmCreat tbsmCreateDataFetcher, tbsmViewDefinitionAdmin, ncw_gauges_editor, tbsmEditTemplate, tbsmViewTemplate, tbsmSLAChartViewVisible, tbsmCreateDataSc
tbsmadmin	Not Active	tbsmadmin	tbs	tbsmLaunchISMServiceReportViewer, iscadmins, tbsmDataFetcherAdmin, ncw_user, tbsmEditDataSource, tbsmEditChart, tbsmViewService, tbsmTemplateAdı tbsmViewChart, tbsmChartAdmin, ncw_admin, administrator, tbsmServiceAdmin, tbsmAVSaveCanvasLayoutForGroup, chartCreator, chartViewer, tbsmAVSaveC tbsmViewDataFetcher, suppressmonitor, tbsmDataSourceAdmin, netcool_rw, tbsmCreateService, tbsmViewRawEvents, monitor, chartAdministrator, tbsmCreat tbsmCreateDataFetcher, tbsmViewDefinitionAdmin, ncw_gauges_editor, tbsmEditTemplate, tbsmViewTemplate, tbsmSLAChartViewVisible, tbsmCreateDataSc



Tree Table

Select a Dataset

Enter a term and press Search to see datasets. To see all, just press Search:

topo [Search] [Cancel]

After searching, select a dataset from the search results below to continue:

Provider: TBSM Service Model > Datasource: TBSM Topology

TBSM Business Impact
Collection of services and relationships for parent services impacted by selected origin services. For use with the Topology widget; Template metrics/KPIs not available.

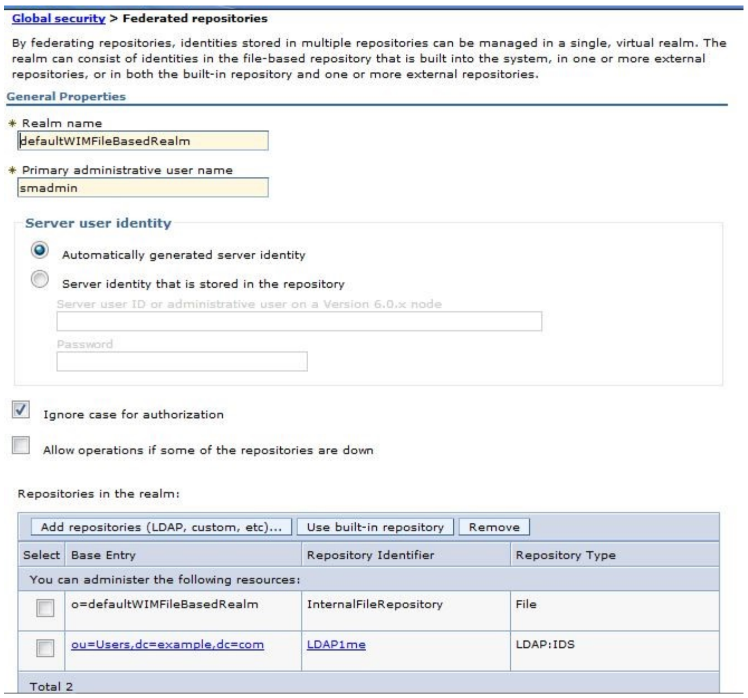
TBSM Topology
Collection of basic service data plus relationships for the Topology and Tree Table widgets; Template metrics/KPIs not available

Provider: Impact_TBSMCLUSTER > Datasource: EIC_alertsdb

EIC_TopologyVisualization
Datatype from Datasource: EIC_alertsdb, name of the datatype: EIC_TopologyVisualization , with table name: alerts.status

Datasets Found: 3

Message: 1 - ⊗:1



Global security > Federated repositories

By federating repositories, identities stored in multiple repositories can be managed in a single, virtual realm. The realm can consist of identities in the file-based repository that is built into the system, in one or more external repositories, or in both the built-in repository and one or more external repositories.

General Properties

* Realm name: defaultWIMFileBasedRealm

* Primary administrative user name: smadmin

Server user identity

- [✓ Automatically generated server identity]
- Server identity that is stored in the repository
  - Server user ID or administrative user on a Version 6.0.x node
  - Password

✓ Ignore case for authorization

Allow operations if some of the repositories are down

Repositories in the realm:

Select	Base Entry	Repository Identifier	Repository Type
You can administer the following resources:			
☐	o=defaultWIMFileBasedRealm	InternalFileRepository	File
☐	ou=Users,dc=example,dc=com	LDAP1me	LDAP:IDS
Total 2			

LDAP:
Using apacheds:

```
[root@adjoint1 apacheds]# ldapsearch -h adjoint1.fyre.ibm.com -p 10389 -x -b 'ou=Users,dc=example,dc=com'
# extended LDIF
#
# LDAPv3
# base <ou=Users,dc=example,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# tbsmAdminUser, Users, example.com
dn: cn=tbsmAdminUser,ou=Users,dc=example,dc=com
cn: tbsmAdminUser
objectClass: top
objectClass: groupOfNames
member: cn=tbsmadmin,ou=Users,dc=example,dc=com

# impactAdminUser, Users, example.com
dn: cn=impactAdminUser,ou=Users,dc=example,dc=com
cn: impactAdminUser
objectClass: top
objectClass: groupOfNames
member: cn=impactadmin,ou=Users,dc=example,dc=com

# tbsmUsers, Users, example.com
dn: cn=tbsmUsers,ou=Users,dc=example,dc=com
cn: tbsmUsers
objectClass: top
objectClass: groupOfNames
member: cn=tbsmadmin,ou=Users,dc=example,dc=com
member: cn=impactadmin,ou=Users,dc=example,dc=com

# Users, example.com
dn: ou=Users,dc=example,dc=com
ou: people
ou: Users
objectclass: top
objectclass: organizationalUnit

# impactadmin, Users, example.com
dn: cn=impactadmin,ou=Users,dc=example,dc=com
sn: impactad
cn: impactadmin
objectclass: top
objectclass: inetOrgPerson
objectclass: person
objectclass: organizationalPerson
description: A impact admin user
userpassword:: e1NTSEF9cDEwNjNWcjNpbGRPUXRpMEZIOGZmRktXRmllWWQrRWMrOHFlbmc9PQ=
 =
uid: impactadmin

# tbsmAdmins, Users, example.com
dn: cn=tbsmAdmins,ou=Users,dc=example,dc=com
cn: tbsmAdmins
objectClass: top
objectClass: groupOfNames
member: cn=tbsmadmin,ou=Users,dc=example,dc=com
member: cn=impactadmin,ou=Users,dc=example,dc=com

# tmpusr, Users, example.com
dn: cn=tmpusr,ou=Users,dc=example,dc=com
sn: tmp
cn: tmpusr
objectclass: top
objectclass: inetOrgPerson
objectclass: person
objectclass: organizationalPerson
description: A temp admin user
userpassword:: e1NTSEF9Y2VhcjUxaFlaUk1XNTRkcFBQRDJ1UDlPdjg2VzVma0R6Rk9nQXc9PQ=
 =
uid: tmpusr

# tbsmadmin, Users, example.com
dn: cn=tbsmadmin,ou=Users,dc=example,dc=com
sn: tbs
cn: tbsmadmin
```

```
objectclass: top
objectclass: inetOrgPerson
objectclass: person
objectclass: organizationalPerson
description: A tbsm admin user
userpassword:: e1NTSEF9aW5XcmtqbkVaQ2hQWC9sS2ZNMmpDaWM0SndZajk5cGlrV08weXc9PQ=
 =
uid: tbsmadmin

# search result
search: 2
result: 0 Success

# numResponses: 9
# numEntries: 8
```

followed the links in:
- https://www.ibm.com/support/knowledgecenter/SSSPFK_6.2.0/com.ibm.tivoli.itbsm.doc/tip/ttip_config_ldap.html
- https://www.ibm.com/support/knowledgecenter/SSSPFK_6.2.0/com.ibm.tivoli.itbsm.doc/tip/ttip_config_ldap_config.html
- https://www.ibm.com/support/knowledgecenter/SSSPFK_6.2.0/com.ibm.tivoli.itbsm.doc/tip/ttip_config_ldap_users.html

Log in error seen after a user is logged out due to inactivity

Problem: If user is logged out due to inactivity, while logging in again may see this error: "Error - Tivoli Business Service Manager <p><p>An error occurred while making the server request.<p><p><b>Error:</b> expected expression, got '<' "

Solution: Close the current tab, open new tab and login again. If you are still not able to login, clear the browser cache and login.

Need to use different directory for APM and TBSM dbconfig (TBSM XMLToolkit & APM)

Problem:

This was the issue, the customer had a remote Db2 server. They installed the APM databases on it then later installed the TBSM database on it. Both products have Java UDF's that are added to their respective databases. TBSM has a database configuration tool for setting up its database, APM has a similar tool. They both default the installation path to ..../tivoli/tbsmdb.

Solution:

Note for users installing TBSM XMLToolkit and APM on the same remote DB2 server.
Both TBSM XMLToolkit and APM and use Java UDFs in DB2. The jar files are cataloged at the server level (not the database level). If both TBSM and APM are installing the jars with the same DB2 name, the jar installed last replaces the one installed first. Since the TBSM jar is a subset of the functions defined in the APM jar, some of APM functions will be missing and APM will not be able to work.
If you run the TBSM Database Configuration Tool after APM, when it has run you must then run the following script from the APM configuration tool to copy back the jars from APM back into DB2:
tbsm_db.sh -s sc -U db2inst1 -f j
If you fail to run the script, some APM functions will be missing and APM will be unable to run.
For details of the APM interim fix related to this issue, see http://ibm.biz/APMandXMLToolkitUDFconflict

No children on expanding Imported Business Services

Symptom: Even though the CRViewer shows components imported from TADDM, there are no services displayed when the Imported Business Services is Expanded in TBSM Service Tree.

Cause: The DB2 driver used by Impact was upgraded to v4 by APAR IJ12231 in FP16. This results in the SQL from the ESDA policies returning incorrect column names. This is because of a change in how aliases are handles in v4 DB2 driver. https://www-01.ibm.com/support/docview.wss?uid=swg21975352

The following is the SQL run via JDBC for ESDA top level services (this can be seen in TBSM_policylogger_ESDA_Custom_SCC_TOPLEVEL_Down_1.log)

```
select distinct sc.id as "id",  sc.cntDepends as "dependency_cnt",   sc.class  as "class",  sc.label as "label", sc.radinstanceid as "radinstanceid", sc.primaryTemplate as "primarytemplate", sc.otherTemplates  as "othertemplates"
from tbsmscr.view_componentswithtemplates as sc   LEFT OUTER JOIN tbsmscr.view_dependencycomponents o  on o.tgtid=sc.id and o.srcprimarytemplate IN  (
'BSM_SOProcess','BSM_BusinessService','BSM_BusinessApplication' )  LEFT OUTER JOIN  tbsmscr.view_tbsmcreatedobjs  as tbsmobjs ON tbsmobjs.comp_id=sc.id  where  o.srcid is null and tbsmobjs.comp_id is null and
sc.primarytemplate IN ( 'BSM_SOAProcess','BSM_BusinessService','BSM_BusinessApplication' ) WITH UR
```

With v3 DB2 driver the columns returned are ***id,dependency_cnt,class,label,radinstanceid,primarytemplate,othertemplates***

With v4 DB2 driver the columns returned are ***ID,CNTDEPENDS,CLASS,LABEL,RADINSTANCEID,PRIMARYTEMPLATE,OTHERTEMPLATES***

This change in column name causes an issue for the ESDA policies in TBSM.

Resolution:

Ensure the only DB2 driver in impact/lib3p or impact/dsalib is v4. db2jcc4.jar.
Create a file TBSM_com.ibm.db2.jcc.DB2Driver.props under $IMPACT_HOME/etc and including the property :
useJDBC4ColumnNameAndLabelSemantics=2
This does require a restart of the Impact server.
Invalidate the Imported Business Services top level service and then re-expand the Service.
If necessary, run a TADDM import, or import an IDML book.
Ensure the services are created in TBSM.

Note- there is a related issue which happens if TBSM 620 GA is used when Impact is at version FP14 or higher. Because Impact FP14 moved to java 8 and TBSM 620 is at an earlier Java version, the following error will be seen in the toolkit logs after import from TADDM :

```
[2019/06/17-16:05:53.089] com.ibm.tbsm.cltools.service.ASIRadFacade getFacade [34] GTMCL5309I: Initiating connection with the TBSM facade at noi09.noi.local using port 9080
[2019/06/17-16:05:58.312] com.ibm.tbsm.cltools.service.ASIRadFacade getFacade [34] GTMCL5342E: An error occurred while invalidating resources in the TBSM Data server. This a permanent error, therefore invalidation will be disabled. Other aspects of data processing will continue as normal, but new data will not appear on the TBSM UI unless the effected resources are invalidated from the TBSM UI. This error is often caused by a classpath issue. The property DL_TBSM_Server_Classpath in xmltoolkitsvc.properties contains the jar files used for invalidation. This property should be changed so that it includes the missing files. Once the classpath has been corrected, the toolkit will need to be restarted to pickup the change. Invalidation will automatically be reactivated when the toolkit is started.
[2019/06/17-16:05:58.312] com.ibm.tbsm.cltools.service.ASIRadFacade getFacade [34] GTMCL5205E: Exception caught. JVMCFRE003 bad major version; class=com/micromuse/response/dblayer/ConfigRepository, offset=6.
```

The above means that the toolkit cannot tell TBSM to invalidate its tree which means the ESDA policies will not run. The fix is to install TBSM 620 FP1. The workaround is to enable the invalidate button in the TBSM UI for Imported Business Services. Click the button and then expand the tree.

Roles For TBSM

By Default tbsmAdmin user will have all necessary roles.

For others users:

add role tbsmAdminUser to see the TBSM Icons on Right Hand Side of DASH Menu

add role tbsmTemplateAdmin and tbsmServiceAdmin to see all templates and Services.

For blank Service Editor / Error in Editing a Service (Service Editor)

Error 500: java.lang.RuntimeException: Could not initialize soap stub for user: smadmin: CTGBF0012E An error occurred while establishing a session. The user ID smadmin is not valid.

Add smadmin user to whatever repository is used for Impact. i.e. add new Omnibus user smadmin with tbsmAdmin/tbsmUsers role.

Generally, if you want another user to be able to access the Impact console, the user will need to have roles in Impact. See here for adding roles in Impact:

https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.14/com.ibm.netcoolimpact.doc/admin/mapping_users_to_roles.html

To see the roles that tbsmadmin has, run the command: $IMPACT_HOME/install/security/mapRoles.sh -list -user tbsmadmin

You should see:

```
tbsmadmin roles:
   bsmAdministrator
   impactAdminUser
```

However, the user smadmin is usually defined in the local file based repository on JazzSM/DASH and therefore will not be accessible to Impact. You can only access the integrated console with users defined in the SSO repository.

One quick word of warning - Websphere will not allow users to exist in more than one repository (the user will become disabled) in the federated set, so please don't be tempted to add smadmin to the OMNIbus repository.

To resolve, I added Object Server user, gerAdmin, with all Groups in Omnibus and Impact roles as follows:

```
./mapRoles.sh -add -user gerAdmin -roles "bsmAdministrator|impactAdminUser"
```

Service Availability CTGTD1000E NullPointerException

Service Tree is blank and error pop up seen

CTGTD1000E
An error occurred while processing the request to the server.
Detail: NullPointerException
-?-

This only occurs in the case where TBSM was installed, the Service Tree Widget was changed to point at a new template, then TBSM was uninstalled and then reinstalled again.

The issue is that the JazzSM is storing the preferences for the TBSM Service Tree Widget, even when the Service Availability Page is deleted (by TBSM uninstall)
Then, when TBSM is installed on the same JazzSM, the preferences are being referenced. Because it is now referencing a template that does not exist, the system gives error.

The workaround is to edit page > edit Widget and to change the tree template to one of the default options. Then save and exit.

Service Details Help button

Problem:
In the Service Administration page, in the Service Details Widget, the help button on the widget toolbar goes to
https://{hostname}:{port}/ibm/help/index.jsp?topic=%2Fcom.ibm.webtop_ua.doc%2Freference%2Fweb_use_lel_overview.html
Which gives a "Topic not found" error.

Solution:
Refer to the relevant documents for help rather than using the Help button on the UI widget.

TBSM 6.1 Data and Dashboard servers separated by a firewall

If you implement theTBSM 6.1 Data and Dashboard servers in a firewall environment with the two servers separated by a firewall, you will find that a connection is attempted from the dashboard server to the data server on a random port and vice versa. If the firewall doesn't allow it, the connection will fail with a mess containing: Connection refused to host: <name or IP addr>

The dashboard server does not initialise correctly. No status or config changes are sent to the Dashboard Server. The following error message is displayed in the Data Server logs.

```
updatepublish 1 com.micromuse.sla.updatepublisher.ClientUpdateHandlerThread run ENTER^ERROR WRITING to client <dashboard host>:17543 we will remove this client updater.
```

```
Connection refused to host: <dashboard host>; nested exception is:
```

```
java.net.ConnectException: Connection timed out
```

Note:

The error message above will always report the port of the rmi registry, even if the communication is failing when running rmi stubs using a different port. This can be misleading as netstat may show an established connection to port 17543, but TBSM is failing to run rmi stubs on a random port. Check the firewall to see what port communication is failing on.

rmi communication typically require at least two server ports - one to lookup stub information in the rmi registry and another to run the stub on the remote server. The rmi registry port is defined with parameter: impact.server.rmiport

On the Data Server this has a default value of 17542 and is stored in $TBSM_HOME/etc/TBSM_server.props

On the Dashboard Server this has a default value of 17543 and is stored in <INSTALL_DIR>/tipv2/profiles/TIPProfile/installedApps/TIPCell/isc.ear/sla.war/etc/RAD_server.props

By default, a random port is used when the running rmi stubs on a remote server. This can cause problems when a firewall exists between the servers if the random port is blocked. If a firewall exists between the data server and the dashboard server, then the server port used for running rmi stubs may need to be opened and specified.

To specify this port, on the data server, in the file: $TBSM_HOME/etc/TBSM_server.props create the entries:

impact.rmiPortRangeStart=17544

impact.rmiPortRangeEnd=17544

On the dashboard server, in the file: <INSTALL_DIR>/tipv2/profiles/TIPProfile/installedApps/TIPCell/isc.ear/sla.war/etc/RAD_server.props

impact.rmiPortRangeStart=17544

impact.rmiPortRangeEnd=17544

17544 is a suggested port, you can use a different free port value if you prefer. You can also use a range of more than 1 port if needed by Impact.

Note: this port should be separate from the impact.server.rmiport which is used for the rmi registry.

TBSM 6.2.0 FP1 Install Failures

2 known issues

----

1) Issue occurs when the base release from July is used instead of the November release.

Symptom:

FP1 fails to install. Error is:

" No metadata found for installed package com.ibm.tivoli.tbsm.dashserver.6.2.0.20181106_0249 com.ibm.cic.agent.internal.core.InstallRegistry"

Resolution

The Readme for FP1 mentions this issue but further steps are required. See: see https://www-01.ibm.com/support/docview.wss?uid=ibm10886187&aid=1

----

2) Issue occurs if you have tbsmdb and tbsm dataserver on the same host:

Symptom:

FP1 fails to install. The installationscript does this:

```
# Prior to running the installer, update the Tbsm configure and health_chec files
TbsmInstallLocations()
{
   #echo in TbsmInstallLocations
   #echo $imLocation/eclipse/tools/imcl
   TbsmHome=`$imLocation/eclipse/tools/imcl listInstallationDirectories -long | grep ": IBM Tivoli Business Service Manager :" | awk '{print $1}'`

   echo "The TbsmHome is: $TbsmHome"
}
```

and gets this:

```
[root@noi09 tools]# ./imcl listInstallationDirectories -long
* : /opt/IBM/IBMIMShared
/opt/IBM/tivoli/tbsmdb : IBM Tivoli Business Service Manager :
/opt/IBM/tivoli/impact : IBM Tivoli Netcool Impact :
/opt/IBM/tivoli/tbsm : IBM Tivoli Business Service Manager_1 :
```

so finds the tbmsdb directory instead.

Resolution

Plan to fix in FP2. Workaround under investigation.

TBSM 620 GA Download and Refresh

The correct part numbers for TBSM 620 GA are:

CNXG3ML.zip IBM Tivoli Business Service Manager V6.2 Linux 64-bit Multilingual

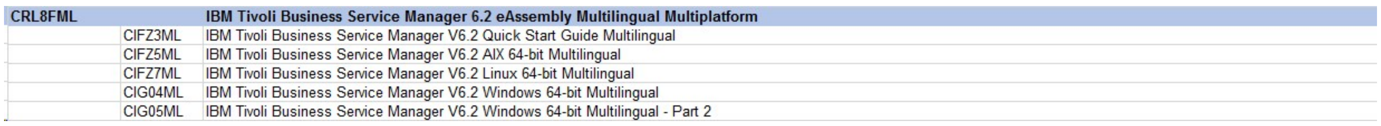
CNXG4ML.zip IBM Tivoli Business Service Manager V6.2 AIX 64-bit Multilingual

CNXG5ML.zip IBM Tivoli Business Service Manager V6.2 Windows 64-bit Multilingual

Note: there was an issue discovered late Nov 2018 where the packages had to be renamed as below:

CNXG3ML IBM Tivoli Business Service Manager V6.2 Linux 64-bit Multilingual	TBSM_V6.2_LINUX_64-BIT_MULTI.zip
CNXG4ML IBM Tivoli Business Service Manager V6.2 Aix 64-bit Multilingual	TBSM_V6.2_AIX_64-BIT_MULTI.zip

For ref: the **old** (July 2018) part numbers are:



CRL8FML		IBM Tivoli Business Service Manager 6.2 eAssembly Multilingual Multiplatform
	CIFZ3ML	IBM Tivoli Business Service Manager V6.2 Quick Start Guide Multilingual
	CIFZ5ML	IBM Tivoli Business Service Manager V6.2 AIX 64-bit Multilingual
	CIFZ7ML	IBM Tivoli Business Service Manager V6.2 Linux 64-bit Multilingual
	CIG04ML	IBM Tivoli Business Service Manager V6.2 Windows 64-bit Multilingual
	CIG05ML	IBM Tivoli Business Service Manager V6.2 Windows 64-bit Multilingual - Part 2

TBSM 620 Policy launch fails from Rule Edit

There are two possible root causes for this issue, each with a different solution.

1: An error appears when a user selects to edit a policy when creating or editing a TBSM Template Rule. The following error occurred while initializing the Policy Editor:

Unable to load https://data-server.rtp.raleigh.ibm.com:9081/restui/policyui/policy/NumericAttributeFunctions/loadPolicyOrTemplate?policyName=NumericAttributeFunctions&template=null status: 0

Solution: First set up Single Sign On between the Impact servers and the Jazz SM server. The steps for this are available in the Impact Knowledge Center here:

https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.13/com.ibm.netcoolimpact.doc/admin/imag_configure_single_signon.html

Once SSO is enabled, enter this url in a new tab (replacing BACKENDHOST)

https://BACKENDHOST:9081/restui/policyui/policy/NumericAttributeFunctions/loadPolicyOrTemplate?policyName=NumericAttributeFunctions&template=null

Then **accept the certificates in the browser** and the Policy Editor should be accessible.

2: CORS security browser error is blocking launch i.e. If I the URL is launched directly, the policy returned ok.

https://BACKENDHOST:9081/restui/policyui/policy/NumericAttributeFunctions/loadPolicyOrTemplate?policyName=NumericAttributeFunctions&template=null status: 0

but clicking Edit policy from the DASH UI, shows Error/Warnings in the Console tab of the browser - related to CORS: Request was blocked. To verify this, open the Debugger on the browser window where the Editor is to be launched from. Check to see if any warnings are shown about CORS Cross Origin Request Blocked.

To resolve the issue stop the Impact backend cluster member. This is the cluster member called TBSM. Edit the server.xml file i.e.

/opt/IBM/tivoli/impact/wlp/usr/servers/TBSM/server.xml

and add the base URL to the list expected (replacing data-server.rtp.raleigh.ibm.com) :

<cors domain="/restui" allowedOrigins="**https://dash-server.rtp.raleigh.ibm.com:16311**" allowedMethods="GET, DELETE, POST, OPTIONS, HEAD" allowedHeaders="x-requested-with, content-type, authorization" allowCredentials="true" maxAge="3600"/>

Restart the back end server.

See also "**The Policy Editor gives exception when accessed from TBSM Rule editor**" https://www-01.ibm.com/support/docview.wss?uid=ibm10716855

TBSM Charting gives error and fails to render chart

Problem : TBSM Charting gives the error below and does not work

Error collecting data visualization input data.
Failed creating dataset.
ATKRST132E An error occurred while transferring a request to the following remote provider: 'Impact_TBSMCLUSTER.provider'. Error Message is '404:Cannot access data provider - Impact_TBSMCLUSTER.provider'.

Solution : This is an known issue with Impact and JazzSM interaction. See the link below for solution
http://ibm.biz/Networkandconnectivityissues

TBSM Data Server won't uninstall when Impact Data Server is not running

Problem: TBSM Data Server won't uninstall when Impact Data Server is not running

Solution: If possible, start Impact.
If Impact cannot be started or repaired, please contact the support team

TBSM Oracle Data Fetcher can't handle hash characters

Problem:

One of the columns of the table in Oracle Data Source is called PHONE#

If the query is:
select ADDRESS,AGE,CITY,CLIENTID,FIRST_NAME,GENDER,LAST_NAME,PHONE# from CLIENTS where AGE > 20 order by CITY

Below error generated:
CTGBA0018E
Verify that the query is valid. CTGBA0049E An exception occurred while processing the query select ADDRESS,AGE,CITY,CLIENTID,FIRST_NAME,GENDER,LAST_NAME,PHONE. Syntax, Duplicate or Resync Error...

Solution:

In order to include the hash character (#) in Oracle queries, you need to set the
db.tableandfield.illegalchars property in the server.props file. This property
specifies which characters are illegal within Oracle queries.
By default, this property is set to "-|#|%|\\+|/|\\s+", which includes # among the
characters deemed illegal. If # is included within the name of a column (for
example: PHONE#), and you want to include that column within an Oracle query
without getting an invalid character error, you need to change the value set for the
db.tableandfield.illegalchars property to "-|%|\\+|/|\\s+". This ensures that #
is no longer treated as an illegal character.

TBSM PMR Data Collection

In a continuing effort to improve the data collection, processing and turn around times associated with troubleshooting TBSM related issues; some changes are being implemented (both on the front-end/customer side and back-end/IBM PMR analysis). Part 1 of the changes, is the collection process. Part 2, is the analysis of the data collected.

As part of those changes (and continuing efforts), on the front-end, the collect_logs utility has been updated to allow for a greater range of data collection. On the back-end, when the data collected via the collect_logs is sent to IBM, the PMR will automatically be stamped with the version info (for all involved components) and any additional findings. Additional logic has been added to the code, to detect the presence of core and javacore files on the system. If javacore files are detected, those will automatically be collected. Core files, if detected, will not be collected (as these require additional processing) but, will be flagged and noted on the PMR text (for the Level 2 Support team to follow-up on with customer).

The changes to the data collection (expansion of data collected and additional logic) will help in improving the collection process (gathering all of the logs, trace and some property files in a single step). Collecting all of the data at once, will hopefully prevent the need to go back multiple times for logs (that by that time, are out of date and might not contain the needed information).

On the analysis side (back-end), when the data arrives at IBM, it will be automatically scanned as part of the "Phase 1 Problem Determination" process. This process will scan all the available logs and other files provided with a PMR, and make an inventory of all the "anomalies" that it finds among all these files (error messages, warning messages, exceptions). The "Phase 1 Problem Determination" process will automatically perform searches for matches for these anomalies or "symptoms" in a variety of Knowledge Bases, to attempt to identify possible solutions or explanations for them. This information will be made available to the Level 2 Support team and should help in quickly finding possible solutions to the customer. A fully analysis will also ensure that any other issues the customer has not yet noticed, are also addressed.

In order to take advantage of the above improvements, customers are being asked to download and start using the new collect_logs tool. Information to download and install this tool, can be found under the following Technote:

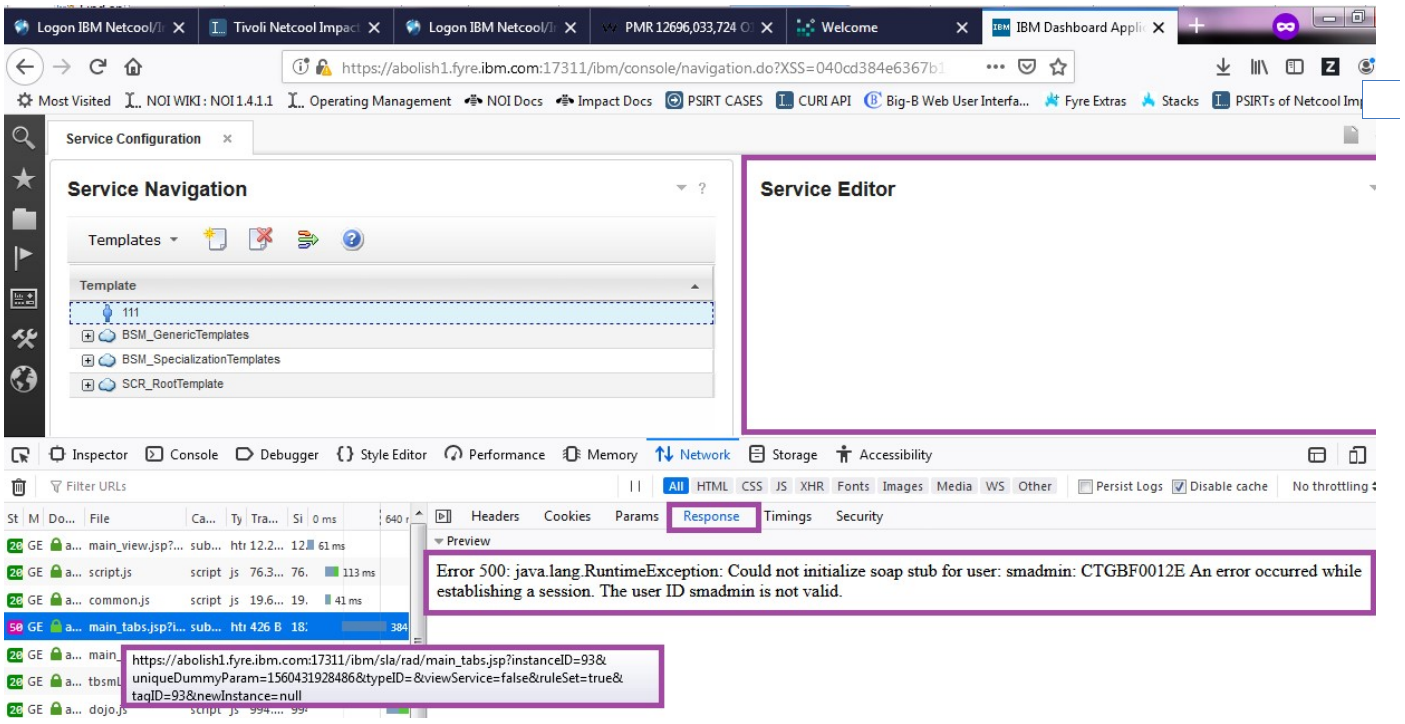
http://www-01.ibm.com/support/docview.wss?uid=swg21598358

TBSM Service Editor is blank

I know 2 reasons why you might see this

1. Impact upgrades removes TBSM war files. This is fixed as per FP16 of Impact. Section **Upgrading Impact removes TBSM war file** here

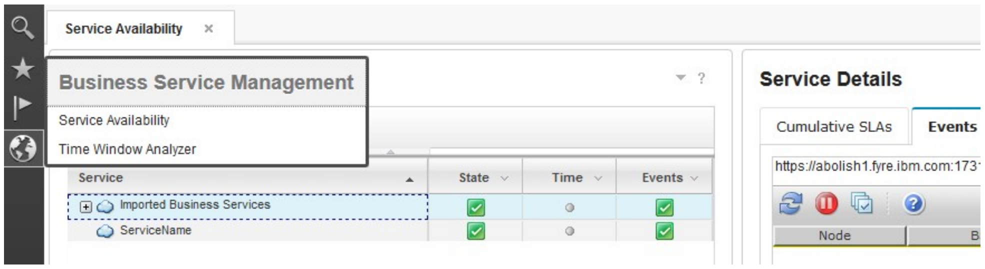
2. The issue happens if the user logged into DASH does not exist in the user repository used by Impact. By rights, both Impact and DASH should be using the same repository so the situation below should not happen. But if, for example, DASH was using filed based + Omnibus federated repository, and Impact was using only file based, then the situation below could happen. To check the repository used for DASH check the Security setting in Websphere Administration Console. To check the repositories used in Impact, look for Omnibus or LDAP added to the features.xml and also their corresponding xml file. To resolve the below I added Object Server to impact :
./confAuth4OMNIbus.sh enable impactadmin impactpass "" impactpass



https://abolish1.fyre.ibm.com:17311/ibm/sla/rad/main_tabs.jsp?instanceID=93&uniqueDummyParam=1560431928486&typeID=&viewService=false&ruleSet=true&tagID=93&newInstance=null

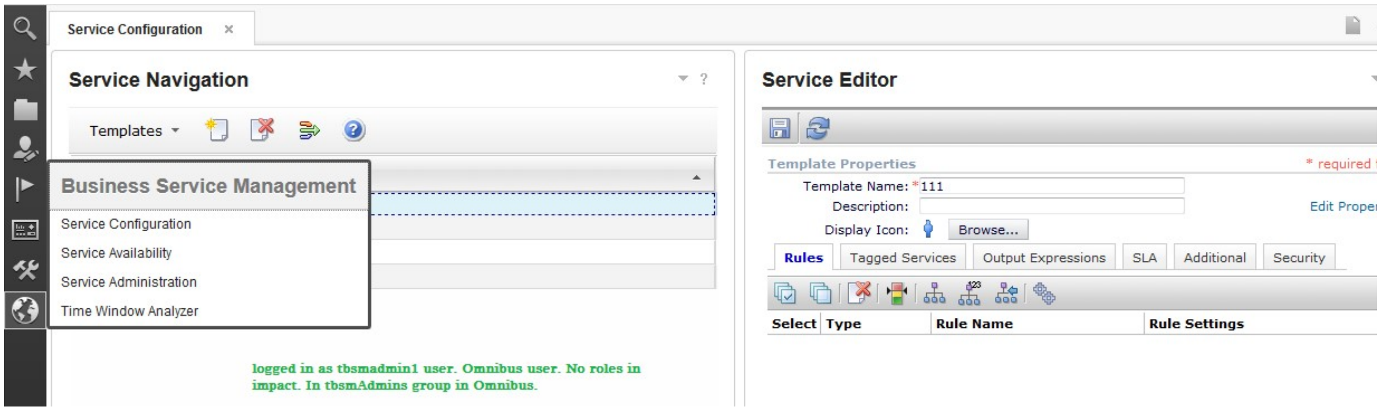
Error 500: java.lang.RuntimeException: Could not initialize soap stub for user: smadmin: CTGBF0012E An error occurred while establishing a session. The user ID smadmin is not valid.

When an Omnibus (or LDAP) user is in the tbsmUsers group in Omnibus (or LDAP) but not in tbsmAdmins this is the menu shown.



When an Omnibus (or LDAP) user is in the tbsmAdmins group in Omnibus (or LDAP) this is the menu shown. No roles needed on Impact.

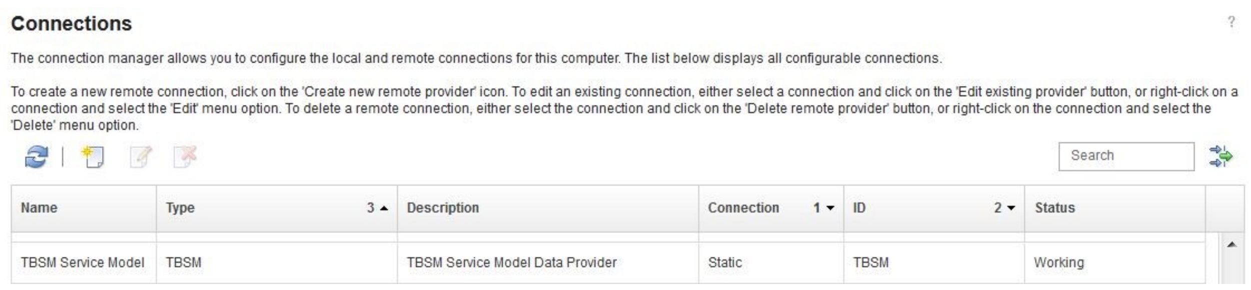
```
abolish1:/opt/IBM/tivoli/impact/install/security # ./mapRoles.sh -list -user tbsmadmin1
Unable to locate any user roles.
```



logged in as tbsmadmin1 user. Omnibus user. No roles in impact. In tbsmAdmins group in Omnibus.

TBSM Service Model Data Provider Test

Below show the TBSM Service Model connection in Dash:



Connections

The connection manager allows you to configure the local and remote connections for this computer. The list below displays all configurable connections.

To create a new remote connection, click on the 'Create new remote provider' icon. To edit an existing connection, either select a connection and click on the 'Edit existing provider' button, or right-click on a connection and select the 'Edit' menu option. To delete a remote connection, either select the connection and click on the 'Delete remote provider' button, or right-click on the connection and select the 'Delete' menu option.

Name	Type 3 ▲	Description	Connection 1 ▾	ID 2 ▾	Status
TBSM Service Model	TBSM	TBSM Service Model Data Provider	Static	TBSM	Working

If the above connection is not displayed on the Connection page, then the below URL can be entered to test whether the data provider is working on Dash:

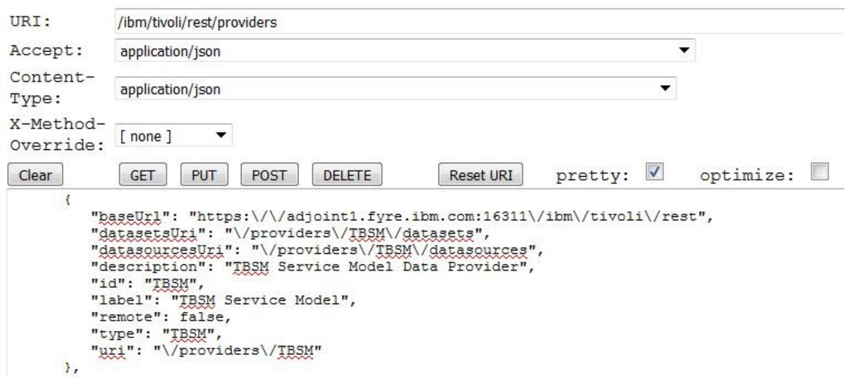
https://Dash_server_hostname:port/ibm/tivoli/test.html

for example: https://adjoint1.fyre.ibm.com:16311/ibm/tivoli/test.html

Enter the URI as /ibm/tivoli/rest/providers

Click "GET"

Below is a screenshot:



URI: /ibm/tivoli/rest/providers
Accept: application/json
Content-Type: application/json
X-Method-Override: [none]
pretty: ✓ optimize:

```
{
    "baseUrl": "https:\/\/adjoint1.fyre.ibm.com:16311\/ibm\/tivoli\/rest",
    "datasetsUri": "\/providers\/TBSM\/datasets",
    "datasourcesUri": "\/providers\/TBSM\/datasources",
    "description": "TBSM Service Model Data Provider",
    "id": "TBSM",
    "label": "TBSM Service Model",
    "remote": false,
    "type": "TBSM",
    "uri": "\/providers\/TBSM"
},
```

The response data should contain an item as shown above, for the "TBSM Service Model Data Provider".

TBSM Split install post installation steps

Problem : When installing TBSM on a system when Impact Server and Impact GUI are installed on separate machines there are issues with Charting and some Impact pages being viewed from TBSM.

Solution : TBSM system status page

1) Edit /opt/IBM/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc/RAD_server.props and set these three fields for remote Impact GUI

```
impact.gui.host=<ImpactGUI server hostname >
impact.gui.httpsport=<ImpactGUI server HTTPS port>
impact.gui.httpport=<ImpactGUI server HTTP port>
```

2) Add tbsmadmin to ImpactFullAccessUser role as shown in bold below by editing the file <impact server install dir>/wlp/usr/servers/ImpactUI/server.xml

```
<!-- ImpactUI EAR -->
  <application type="ear" id="Impactui" name="ImpactUI" location="${server.config.dir}/apps/ImpactUI.ear">
    <classloader privateLibraryRef="lib3p, uilib, wlplib" apiTypeVisibility="spec, ibm-api, third-party"/>
    <application-bnd>
      <security-role name="impactAdminUser">
        <user name="${impact.user}"/>
      </security-role>
      <security-role name="impactFullAccessUser">
        <user name="${impact.user}"/>
        <user name="tbsmadmin"/>
      </security-role>
```

3) Restart Jazz SM

TBSM Charts

1) Open the Connections page in JazzSM-> Console Settings -> Connections
2) Navigate to Impact_TBSMCLUSTER connection
3) Edit connection by selecting connection and clicking pencil icon to specify Impact GUI server fqdn as ' Host name' and Impact GUI HTTPS port as 'Port'
4) Click search button and it'll list Impact_TBSMCLUSTER as shown below.
5) Select radio button next to Impact_TBSMCLUSTER and click ok.
6) Restart Jazz SM

TBSM UI Server fails to install on JazzSM FP3

JazzSM has committed to delivering a fix in FP5 (Sept/Oct) to resolve this issue. APAR is **IJ17703**.

The issue is that the restcli.sh command shipped with JazzSM FP3 requires Java 8.

Depending on the order of Websphere and Java installs the default Java version may be older than 8. In this case the restcli command called by TBSM install will fail. The TBSM install fails cleanly. The workaround is to use Java 8 by updating JAVA_HOME for restcli.sh on this server.

The workaround is documented here: http://www.ibm.com/support/docview.wss?uid=ibm10888471

Note: Java 8 was shipped with Websphere starting 8.5.5.14.
With 8.5.5.9 the java version is version 6. There is a Websphere utility for switching between Java versions.
If you install Websphere and Java at the same time then the default version used by Websphere will be the new Java version which was installed at the same time. However, if Java is installed post Websphere then the default java version used by Websphere will not change.

For information of Java 8 with websphere see:
https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/covr_javase8.html

Reference: Salesforce TS002372393 Error while installing TBSM6.2 DASH (Using New Build)

Error is
[ncosys@DRV2TBSMDASH native]$ cat 20190614_1945e.log

```
Exception in thread "main" java.lang.UnsupportedClassVersionError: JVMCFRE003 bad major version; class=com/ibm/tivoli/rest/cli/RestCLI, offset=6
at java.lang.ClassLoader.defineClassImpl(Native Method)
at java.lang.ClassLoader.defineClass(ClassLoader.java:273)
at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:74)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:563)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:474)
at java.net.URLClassLoader.access$300(URLClassLoader.java:77)
at java.net.URLClassLoader$ClassFinder.run(URLClassLoader.java:1059)
at java.security.AccessController.doPrivileged(AccessController.java:488)
at java.net.URLClassLoader.findClass(URLClassLoader.java:452)
at java.lang.ClassLoader.loadClassHelper(ClassLoader.java:701)
at java.lang.ClassLoader.loadClass(ClassLoader.java:680)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:358)
at java.lang.ClassLoader.loadClass(ClassLoader.java:663)
Could not find the main class: com.ibm.tivoli.rest.cli.RestCLI. Program will exit.
```

Test Page

The data fetcher is locked by user tbsmadmin

Problem:
The data fetcher is locked by user "tbsmadmin", no data fetcher can be created by other users.

If an input strings 'ABC???Çàâ?????????????€¥?' {non-english characters) is entered on the data fetcher name field by user "tbsmadmin", it will be locked by user tbsmadmin.

When another user is to create a new data fetcher, it will fail with message "locked by user tbsmadmin".

Below error is generated in TBSM Data Server log file:
CTGBA0007E An exception occurred while saving the data fetcher named ABCÃÃ Ã¢â¬Â¥. Version Control System [SubVersion] add of /opt/IBM/tivoli/impact/etc/TBScÃ§Ã Ã¢â¬Â¥.props failed. Error while executing: '"/opt/IBM/tivoli /impact/platform/linux/svn/bin/svn" add "etc/TBSM_abcÃ§Ã Ã¢â¬Â¥.props"'

Solution:
Input English strings on Data Fetcher name again using user tbsmadmin, the files locked by tbsmadmin will be unlocked. Then other users will be able to create new Data Fetcher.

The Policy Editor gives exception when accessed from TBSM Rule editor

Problem: An error appears when a user selects to edit a policy when creating or editing a TBSM Template Rule

Solution: First Single Sign On between the Impact servers and the Jazz SM server must be enabled. The steps for this are available in the Impact Knowledge Centre here:

https://www.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.13/com.ibm.netcoolimpact.doc/admin/imag_configure_single_signon.html

Once SSO is enabled, the user should enter this url in a new tab

https://<TBSM_Dashboard_Server>:9081/restui/policyui/policy/NumericAttributeFunctions/loadPolicyOrTemplate?policyName=NumericAttributeFunctions&template=null

Accept the certificates in the browser and the Policy Editor should be accessible.

Tivoli Business Manager version 6.1.0 installation fails on Window 2008 R1 Enterprise Server

When you install Tivoli Business Manager version 6.1.0 (TBSM6.1) on a Windows 2008 R1 Enterprise Server (64bit), the TBSM installation program fails.

Symptoms

The installation fails at the step: WebSphere® Application Server Fix pack or Tivoli® Integrated Portal Fix pack installation.

Cause

The WASServiceMsg.dll cannot stop because it is locked by the Windows event log.

Resolution

1. Stop Window service Event log (To stop the service, you set the service to Disabled and reboot the machine)

2. Clean up the failed installation

3. Run the TBSM installation program again.

How to clean up the failed TBSM install on window

1. Uninstall the Deployment Engine. Ensure that all products that use the Deployment Engine have been removed before you uninstall it.

32-bit systems
Change to the C:\Program Files\IBM\Common\asci directory:
64-bit systems
Change to the C:\Program Files (86)\IBM\Common\asci directory:
To uninstall the Deployment Engine, run these commands:
a.run setenv
b. cd bin
c. si_inst -r -f
d. rmdir C:\Program Files\IBM\Common\asci /s

2. Remove the Window services for TBSM6.1: the following service names need to be removed

sc delete "ASICRToolkitSvc" // Remove the service Tivoli BSM Discovery Library toolkit

sc delete "IBMWAS70Service - TBSMProfile_Port_17310" // Remove the service Tivoli Bussiness Service Manager - TBSMProfile_Port_17310

sc delete "IBMWAS70Service - V2.2_TIPProfile_Port_16310" // Remove the service Tivoli Integrated Portal - v2.2_TIPProfile_Port_16310

sc delete "NCOObjectServer" // Remove the service Netcoll/OMNIbus Object Server

sc delete "NCOProcessAgent" // Remove the service NCO Process Agent

3. Remove installer log files. In the install user's home directory, delete all TBSMInstall*.log files

4. Remove the TBSM installDir directory

5. Reboot the machine

Topology Widget doesnot display TBSM data

Problem:

The Topology widget does not work on wires all the time. You see number of objects/relations changing at the bottom, but nothing is displayed.

A page has a Tree Table widget with TBSM Services and a topology widget with TBSM Topology, a wire connecting the two widgets. When Tree Table service is selected, Topology is not shown the service, a "blank canvas" occurred. The bottom bar in the widget shows the correct detail regarding the number of Resources and Relationships.

Solution:
Delete the non-working page, and create a new identical page to solve the problem.

Turning on Logging for TBSM 620

TBSM 620 Logging

By default, there is no trace logging enabled for TBSM 620 java code. TBSM java code uses Websphere log methods (rather than the log4j used by Impact) . To generate the TBSM trace logs, similar to TBSM 6.1.1, here are the steps:

TBSM Backend (aka Impact server):

For websphere logging in TBSM code, the file to change is:
./wlp/usr/servers/TBSM/server.xml

Change the logging entries to the below.

```
<logging maxFiles="20" maxFileSize="20" consoleLogLevel="AUDIT" copySystemStreams="false"/>
<logging traceSpecification="*=info:com.ibm.tbsm.*=finest:com.ibm.tivoli.twa.marker.*=finest:com.micromuse.*=finest"/>
```

Unfortunately this requires a restart of the Impact backend server, TBSM.

Log files will be as per normal websphere logging: i.e. impact\wlp\usr\servers\logs\trace...

Event Logging:

To trace the processing of events and status changes, a special Event Logging setting can be set for TBSM. This is done by editting the server.xml file (as above) and including EventLogging=FINER in the traceSpecification. i.e.

```
<logging traceSpecification="*=info:com.ibm.tbsm.*=finest:com.ibm.tivoli.twa.marker.*=finest:com.micromuse.*=finest:EventLogging=FINER"/>
```

Note: FINER must be in upper case.

Log files will be as per normal websphere logging: i.e. impact\wlp\usr\servers\logs\trace...

TBSM UI (aka DASH server):

1. Logging for Eebsphere logging in TBSM code is controlled via DASH and can be changed dynamically via the Websphere Application Console. Use the same trace specification as above.

Log files will be as per normal websphere logging: i.e. JazzSM\profile\logs

2. There is also Impact code used in the TBSM DASH UI. i.e. for connections to the name server there is Impact code (nci.jar, ncCommon.jar, ncClient.jar) in the JazzSM server but the logging is log4j which is not enabled.

To enable log4j logging, a impactserver.log4j.properties file needs to be placed in <install dir>/JazzSM/profile/installedApps/JazzSMNode01Cell/isc.ear/sla.war/etc on the Jazz/Dash server.

See attached.

This requires a DASH restart. See attached for example, but note the path is hard coded in the example. Change log4j.appender.NETCOOL.file=/opt/IBM/JazzSM/profile/logs/server1/impactserver.log as appropriate.

Note: this will be in place ootb after FP2 install of TBSM 620: How to get impact log4j logging in DASH UI to activate ? (69266)

TWA does not work

Problem:
When accessing the TWA you can search for services, but when you select one to view on the chart you get the following error:
CTGBH0014E
Could not retrieve information from the metric service. The service cannot be found for the endpoint reference (EPR) http://{hostname}:9080/ChartService/services/QueryService/

Below error is generated in TBSM Data Server log file:
[INFO] ServiceDeployer - The ChartService service, which is not valid, caused java.lang.NoClassDefFoundError: org.apache.axis2.databinding.utils.writer.MTOMAwareXMLStreamWriter

Root Cause:
TWA uses the ChartService. It looks like the ChartService web service package in JazzSM was compiled using an older version of AXIS2, v.1.3.1. Some of the classes from this version have since been deprecated and removed from AXIS2 v.1.6 and later. When the TWA portlet attempts to make a web service request to TBSM 6.2, it fails because TBSM 6.2 and Impact 7.1 are using AXIS2 v.1.6.2 or higher. We need input from Dash support as to whether it's possible to get these classes recompiled using a newer build of AXIS2.

Workaround/Solution:

Parent Case: TS002086535. Skill Case TS002090448. Skill case with DASH is TS002142027

A workaround was attempted by replacing the AXIS2 1.6.2 jars on the Impact Server with version 1.5.6. But Dash's interface classes were compiled with 1.3.1 and unfortunately downgrading TBSM to 1.5 didn't work.

Upgrading Impact removes TBSM war files

Problem:
Using TBSM Administrator page, TBSM Service Editor is blank when a service is selected.

The Impact FP installer removes TBSM war files from the wlp apps directory.

Solution:
The manual fix for this is to stop the Impact back end server and copy the was files back from the backup.

i.e if upgrade was from FP14, then these are the commands

```
cd $IMPACT_HOME
cp -r ./backup/install/server_backup/7.1.0.14/wlp/usr/servers/TBSM/apps/TBSM.ear/tbsm.war wlp/usr/servers/TBSM/apps/TBSM.ear
cp -r ./backup/install/server_backup/7.1.0.14/wlp/usr/servers/TBSM/apps/TBSM.ear/chartService.war wlp/usr/servers/TBSM/apps/TBSM.ear
cp -r ./backup/install/server_backup/7.1.0.14/wlp/usr/servers/TBSM/apps/TBSM.ear/markerWeb.war wlp/usr/servers/TBSM/apps/TBSM.ear
cp -r ./backup/install/server_backup/7.1.0.14/wlp/usr/servers/TBSM/apps/TBSM.ear/META-INF wlp/usr/servers/TBSM/apps/TBSM.ear
```

Then restart the Impact back-end server.

Using ! in tipadmin password (windows)

On Windows, TBSM password encryption is incorrect when passwords contain a ! (exclamation) character. The tipadmin password is encrypted and stored in property files, but the default generated encryption will be incorrect, if the password contains !.

Symptom

The symptoms are as follows.

1. rad_radshell will not connect to the data server. No rad_radshell prompt will be displayed.

2. The TBSM trace logs will contain the following error: com.ibm.ws.wim.adapter.file.was.FileAdapter login com.ibm.websphere.wim.exception.PasswordCheckFailedException: CWWIM4512E The password match failed.

Note: the login via the GUI will complete OK.

Cause

The ! character has special meaning in windows batch programming, and it is removed from the string before the password is sent for encryption. Setting enabledelayedexpansion in bat files ensures the character is only removed at execution time, but it is expanded when passed to other bat files.

Environment

Windows

Diagnosing the problem

Check the tipadmin password for ! character and check for errors (as per Symptom section) in the TBSMProfile trace logs.

Resolving the problem

The following workaround will ensure that the correct encryption is generated for the tipadmin password, if it contains a ! character.

Use tbsm/bin/nci_crypt to generate the encrypted password and escape the ! character with ^, for example to encrypt a password, t!padm!m, use the following

```
nci_crypt "t^!padm^!n"
```

The output must be saved as the value for property impact.server.vmm.admin.password in the following property files:

Dashboard Server
<INSTALL_DIR>\impact\etc\server.props
$TBSM_DASHBOARD_SERVER_HOME\etc\RAD_sla.props

Data Server
<INSTALL_DIR>\tbsm\etc\TBSM_sla.props

The Dashboard and Data Servers must be restarted for the change to take affect.

XMLToolkit Bulk Load does not complete.

Problem(Abstract)

The load does not complete because the DB2 transaction log is full.

Symptom

The **toolkit** logs file shows error as follows:

```
com.ibm.tbsm.cltools.jdbc.ASIJDBCConnection executeUpdate [1] GTMCL5205E: Exception caught. -964 - DB2 SQL error: SQLCODE: -964, SQLSTATE: 57011, SQLERRMC: null.
```

Cause

The DB2 transaction log fills up and prevents the **toolkit** from completing the bulk load.

Environment

TBSM 6.1.0

Diagnosing the problem

Check the **toolkit** logs for DB2 SQL error 57011 or 0964. This indicate a space issue with the database transaction log.

Resolving the problem

To resolve the problem, increase the space available for the DB2 transaction log, with the following DB2 command:

db2 UPDATE DATABASE CONFIGURATION FOR <TBSM_DATABASE_NAME> USING
LOGSECOND 100;

where <TBSM_DATABASE_NAME> is typically TBSM.

Comments (0) Versions (7) Attachments (19) About

There are no comments.

Add a comment

Feed for this page | Feed for these comments

w3-connections.ibm.com

Overview and Planning (tbsm) - Tivoli Netcool/Impact Wiki

1-2 minutes

Overview and Planning

This section of the wiki includes Tivoli Business Service Manager overview and planning information

Systems with NUMA architecture that is running VMWare ESX/ESXi may experience unexpected virtual machine failure that is accompanied by the following error message in the machine's vmware.log file:

*VMware ESX Server internal monitor error *** vcpu-0:ASSERT vmcore/vmm32/platform/common/platform.c:34 bugNr=17332*

To work around this issue, use the vSphere Client to set the "Numa.MonMigEnable" option to "0" as described in the following article:

Understanding the TBSM Common Agent

Follow the link below to a document that provides a deeper understanding of how the TBSM Common Agent works. The document contains the follow topics :

- Overview
- Data Flow Diagram
- Data Collection Scenario
- Stop Start Scenarisos
- Agent Installation Scenarios
- Troubleshooting
- Historical Data Collection

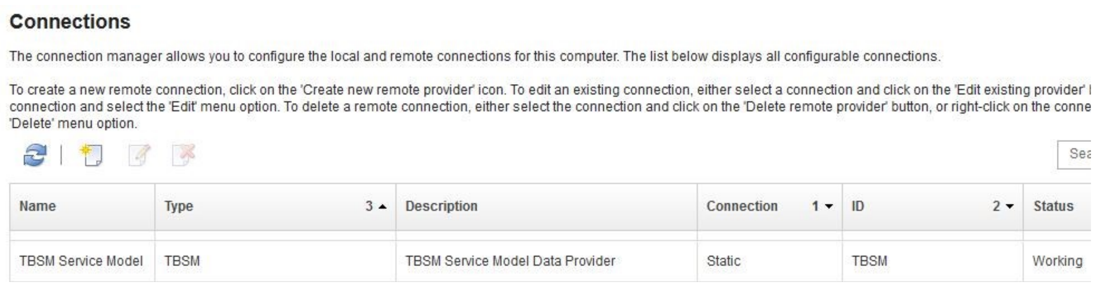
You can find the document here Understanding the TBSM Common Agent

TBSM Service Model Data Provider Test

Like | Updated Oct 18 by HERMAN LO | Tags: *None* Add tags

Edit Page Actions

Below show the TBSM Service Model connection in Dash:



Connections

The connection manager allows you to configure the local and remote connections for this computer. The list below displays all configurable connections.

To create a new remote connection, click on the 'Create new remote provider' icon. To edit an existing connection, either select a connection and click on the 'Edit existing provider' I
connection and select the 'Edit' menu option. To delete a remote connection, either select the connection and click on the 'Delete remote provider' button, or right-click on the conne
'Delete' menu option.

Name	Type 3 ▲	Description	Connection 1 ▼	ID 2 ▼	Status
TBSM Service Model	TBSM	TBSM Service Model Data Provider	Static	TBSM	Working

If the above connection is not displayed on the Connection page, then the below URL can be entered to test whether the data provider is working on Dash:

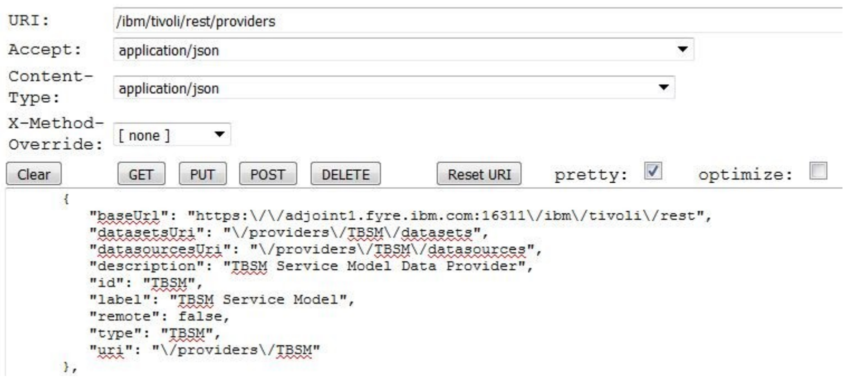
https://Dash_server_hostname:port/ibm/tivoli/test.html

for example: https://adjoint1.fyre.ibm.com:16311/ibm/tivoli/test.html

Enter the URI as /ibm/tivoli/rest/providers

Click "GET"

Below is a screenshot:



URI: /ibm/tivoli/rest/providers
Accept: application/json
Content-Type: application/json
X-Method-Override: [none]

```
      {
         "baseUrl": "https:\/\/adjoint1.fyre.ibm.com:16311\/ibm\/tivoli\/rest",
         "datasetsUri": "\/providers\/TBSM\/datasets",
         "datasourcesUri": "\/providers\/TBSM\/datasources",
         "description": "TBSM Service Model Data Provider",
         "id": "TBSM",
         "label": "TBSM Service Model",
         "remote": false,
         "type": "TBSM",
         "uri": "\/providers\/TBSM"
      },
```

The response data should contain an item as shown above, for the "TBSM Service Model Data Provider".

Comments (0) Versions (1) Attachments (2) About

There are no comments.

Add a comment

Feed for this page | Feed for these comments

Tags

Find a Tag

1.8.5 6.1 7.1 a11y aix angular angularjs apacheds appscan bigb blaze blogs curi dap dash datasource db2 dojo dojobuild eclipse gsa help idx iehs impact java javascript jazzsm jms layer ldap liberty mysql oracle rba rest retain rtc salesforce selenium smc ssl sso tip ttecgo twl vcell webgui wsdl zlinux

Cloud | List

w3-connections.ibm.com

Home (TBSM) - Tivoli Netcool/Impact Wiki

29-37 minutes

IBM Tivoli Business Service Manager 6.2.0
Fix Pack Version 6.2.0-TIV-BSM-FP0001

Readme file for: IBM Tivoli Business Service Manager
Product/Component Release: 6.2.0
Update name: Fix Pack 1
Fix ID: 6.2.0-TIV-BSM-FP0001
Publication Date: April 1, 2019
Last modified date: March 23, 2019
Online version of the readme file:
http://www.ibm.com/support/docview.wss?uid=ibm10876634

Attention: You can always find the most current version of the readme file online.

Contents

==

1.0 Files included in this Fix Pack

Fix Pack 1 addresses the problems that have been reported in IBM® Tivoli® Business Service Manager version 6.2.0. The following table contains a list of files included in this Fix Pack and operating systems associated with these files:

Platform	File
AIX®	6.2.0-TIV-BSM-AIX-FP0001.zip
Linux®	6.2.0-TIV-BSM-LINUX-FP0001.zip
Windows®	6.2.0.-TIV-BSM-WINDOWS-FP0001.zip
all platforms	6.2.0-TIV-BSM-FP0001.README

Note: Whenever <ARCH> is used in the text of this readme, as part of the filename of the Fix Pack package, it refers to, and can be substituted for one of the following operating systems:
* Linux
* AIX
* Windows

The following files have been updated or replaced by this Fix Pack.

Replaced files on the Data Server:
rad_sendevent
rad_sendevent.bat
versioninfo
versioninfo.bat
scriptedAPIStartup.bsh
ncsSoapClient.jar
verutil.jar
xmlschema-core-2.2.4.jar
stax2-api-3.1.4.jar
slf4j-jdk14-1.7.25.jar

serviceinstall.jar
maintcli.jar
jcl-over-slf4j-1.7.25.jar
cxf-rt-transports-http-3.3.0.jar
cxf-rt-databinding-jaxb-3.3.0.jar
cxf-rt-bindings-xml-3.3.0.jar
cxf-manifest.jar
wsdl4j-1.6.3.jar
saaj-api-1.2.jar
neethi-3.1.1.jar
markercli.jar
mail-1.2.jar
cxf-rt-ws-policy-3.3.0.jar
cxf-rt-wsdl-3.3.0.jar
cxf-rt-ws-addr-3.3.0.jar
cxf-rt-frontend-simple-3.3.0.jar
cxf-rt-frontend-jaxws-3.3.0.jar
cxf-rt-bindings-soap-3.3.0.jar
commons-logging-1.0.3.jar
activation-1.0.2.jar
woodstox-core-5.0.3.jar
slf4j-api-1.7.25.jar
jaxrpc-api-1.1.jar
cxf-core-3.3.0.jar
asm-7.0.jar

Replaced Impact files on the Data Server:
av.jar
nciJmxClient.jar
ncs.jar
tbsm-jviews.jar
tbsm-info.jar
markerservice.jar
jdom.jar
jaxb-xjc.jar
jaxb-api.jar
itmdataclient.jar
iscwccm.jar
iscportal.jar
cpci.jar
com.ibm.tivoli.tip.oda.ws.client.jar
tslaAPI.jar
tbsm-jviews-chart.jar
tbsmdata-nls-resources.jar
org.eclipse.emf.ecore.jar
org.eclipse.emf.common.jar
ncsmWebClient.jar
ncsagent.jar
ncChartComponent.jar

markerclient.jar
JSON4J.jar
jaxb-libs.jar
jaxb-impl.jar
icl.jar
com.ibm.tivoli.tip.oda.ws.client.was.jar
bsh-1.3b2.jar
AuditLogger.jar

Updated to java version "1.8.0_201" Java(TM) SE Runtime Environment (build 8.0.5.30)

Replaced files on the Dashboard Server:
av.jar
nci.jar
ncs.jar
versioninfo.bat
ncChartComponent.jar
ncChartClientComponent.jar
ncCommon.jar
nciClient.jar
ncsSoapClient.jar
tbsm-info.jar
CreateDBPoller.jsp
resourceCommon.jsp
splash.jsp
DataSetQueryConfig.jsp
DataSetQueryConfigGetRequiredParams.jsp
ListAllCharts.jsp
SLAChart.jsp
GetCanvas.jsp
PolicyEditorRequire.jsp
TypeEditor.jsp
AuditDojoRequire.jsp
defaultmainpage.jsp
dojoRequire.jsp
ism_server_configuration.jsp
main_viewWrapped.jsp
DataSourceToolbar.jsp
SelectTableJoins.jsp
tools.jsp
toolFooter.jsp
toolHeader.jsp
OslcBusinessImpactPreview.jsp
OslcStatusPreview.jsp
OslcXml.jsp
EventMetricRule.jsp
RawAttributeWizardOption.jsp
SelectDiscriminator.jsp
SelectEventForRawAttribute.jsp

SelectInstanceFieldRawAttribute.jsp
SelectRawAttribute.jsp
SelectThresholdFieldRawAttribute.jsp
raw_event_table_fragment.jsp
chartResourceCommon.jsp
error.jsp
dojoCommon.jsp
timeSeriesCommonBase.jsp
timeSeriesCommonPortlet.jsp
timeSeriesPortletView.jsp
dojo/**
Updated to java version "1.8.0_201" Java(TM) SE Runtime Environment (build 8.0.5.30)

Replaced files associated with the Discovery Library Toolkit:
db2jcc.jar
db2jcc_license_cu.jar
ojdbc6.jar
orai18n.jar
serializer.jar
xalan.jar
xercesImpl.jar
xml-apis.jar
Updated to java version "1.8.0_201" Java(TM) SE Runtime Environment (build 8.0.5.30)

==

2.0 Hardware and software requirements

For information about supported hardware and software, check the Software
Product Compatibility Reports (SPCR) here:
https://www.ibm.com/software/reports/compatibility/clarity/index.html

And the Tivoli Business Service Manager Installation Guide on the IBM
Tivoli Business Service Manager Information Center (at IBM Knowledge Center):

https://www.ibm.com/support/knowledgecenter/SSSPFK_6.2.0/com.ibm.tivoli.itbsm.doc/kc_welcome_tbsm.html

==

3.0 Installation information

3.1 Dependencies

This Fixpack must be installed on each server(s) that is running
the TBSM data server or TBSM dashboard server.

This Fixpack requires IBM Tivoli Business Service Manager
Version 6.2.0.0 to be installed.

3.2 Prerequisites

- Installation Manager on your machine should be 1.8.8 or higher
- JazzSM 1.1.3 Fixpack2 (DASH 3.1.3 FP2)
  Please note: FP2 for JazzSM 3.1.1 must be applied on top of FP1.
  If FP1 was not pre-installed the following workaround
  is required: - https://developer.ibm.com/answers/questions/494209/tbsm-62-tbsm-ui-provider-not-working-properly-if-d.html?childToView=494210#answer-494210)
- Impact 7.1.0 Fix Pack 15

3.3 Special considerations and Notices

----

* TADDM 7.2.0 and 7.2.1 are no longer supported. Refer to the following link for more info: http://ibm.biz/Announcement_of_future_withdrawal_from_support_for_TADDM_7_2_0_and_7_2_1

Consider this information before you install the Fix Pack:

1. Determine the build version of the TBSM 6.2.0.0 package currently on the environment. This can be done running the versioninfo script located in the <TBSM_HOME>/bin directory:

   On Windows:
   For example - "C:\Program Files\IBM\tivoli\tbsm\bin\versioninfo.bat"
   or "C:\Program Files\IBM\tivoli\tbsmdash\bin\versioninfo.bat"

   On Unix (Linux/AIX):

   For example - /opt/IBM/tivoli/tbsm/bin/versioninfo
   or /opt/IBM/tivoli/tbsmdash/bin/versioninfo

   If the utility header output shows a build date of 20180723 (as shown below), perform the actions that follow:

```
=============================================================
TBSM Version Utility:  Build date: 6.2.0. - 201807231316
=============================================================
```

   NOTE: If the utility header output shows a build date of 20181106, you may proceed with the Fix Pack upgrade process.

   a) Return to the site where the original TBSM 6.2.0.0 Installation package was downloaded from and retrieve the updated base packages. Please note that there is NO need to install this new package but, just need to have it accessible. This is only required if for some reason the TBSM 6.2.0 Fix Pack 1 needs to be rolled back, after it is installed!

   The following, are the updated TBSM 6.2.0.0 base install packages with a build date of 20181106:

   CNXG3ML IBM Tivoli Business Service Manager V6.2 Linux 64-bit Multilingual
   TBSM_V6.2_LINUX_64_BIT_MULTI.zip

   CNXG4ML IBM Tivoli Business Service Manager V6.2 AIX 64-bit Multilingual
   TBSM_V6.2_AIX_64_BIT_MULTI.zip

CNXG5ML IBM Tivoli Business Service Manager V6.2 Windows 64-bit Multilingual
TBSM_V6.2_WIN_64_BIT_MULTI.zip

b) Determine location where the original installation package was placed.
This can be done by either of following two methods:
i) Launch the Installation Manger

If using console mode:
- select V. View Installed Packages
- then each of the TBSM Components (Data Server, Dashboard Server)
- on the output, look for the directory shown on the "Repository" field.

If using GUI mode:
- Click "File"
- then click on "View Installed Packages"
- Select each of the TBSM Components (Data Server, Dashboard Server)
- look for the directory shown on the "Repository" filed on the "Details" section.

for example:
Repository: /Downloads/tbsm/output_data

ii) Launch the Installation Manager imcl "command line" utility to list the installed packages. For example:

```
/opt/IBM/InstallationManager/eclipse/tools/imcl listInstalledPackages -verbose
```

Look for the "Repository" information associated with both the Dashboard and Data Server applications:

for example:

```
Name: IBM Tivoli Business Service Manager Dashboard Server (com.ibm.tivoli.tbsm.dashserver)
Version: 6.2.0.0 (6.2.0.20180723_1407)
Repository: /opt/tivoli/tbsm/output_dash
```

c) Backup or rename the repository locations (identified above), for each of the TBSM components (Data and Dashboard Server).

c) Place the unpacked package (retrieved in step a), for each of the TBSM components (Data and Dashboard Server) on the appropriate Repository location (identified above). Basically overlaying the old version (build 20180723) with the new base build version (build 20181106)

2. All non-vital programs should be closed prior to installation of the Fixpack.
This includes the Impact Data and Dashboard servers, as well as, JazzSM (hosting the TBSM Dashboard Server).

3.4 Superseded Fixes

None

3.5 Extracting Fix Pack files

3.5.1 Extracting on AIX, Linux and Windows platforms

1. Copy the file 6.2.0-TIV-BSM-<ARCH>-FP0001.zip to a temporary location on your TBSM Servers.

2. Unzip the file on each server where you need to install the Fix Pack.

3.5.2 Fix pack directories

When you extract the file, these directories are created:

dash_<ARCH>	-TBSM dashboard server updates
data_<ARCH>	-TBSM Data server updates

```
The above listed directories, each will contain the following directories:
  <ARCH> (Contains TBSM Data/Dashboard Server updates)
  scripts (Contains installer scripts and a sub-directory with sampleResponsefiles)
    patch_base (Contains script, for making adjustment between base releases)
    sampleResponseFIles (Contains the <ARCH> specific directory for the response file)
      <ARCH> (Contains the Data or Dash_resp_update_<arch>.xml response file)

  and the following files are available:
   update_console_<arch>.* (Update in console mode)
   update_gui_<arch>.*     (Update in GUI mode)
   update_silent_<arch>.*  (Update in Silent mode)
```

3.6 Maintenance upgrade strategy

Before you install the Fix Pack, you need to plan how and when you will upgrade each TBSM server in your environment.

3.6.1 Example maintenance scenario

In this example, the maintenance is staged into two phases to reduce the time required for the TBSM maintenance window.

In this example environment, there are four servers:
* primary data server
* backup data server
* two dashboard servers set up for load balancing called: dash1 and dash2

Follow this sequence:

Before you start the maintenance window:
1. Upgrade backup data server
2. Upgrade one dashboard server (for example dash2)

3. Use the primary data server and dash1 for regular production activities

After you complete the upgrade the two secondary servers, start the maintenance window down time for the production servers.
1. Upgrade primary data server
2. Upgrade dash1

After you complete the upgrade for all the servers:
1. Start primary data server
2. Start backup data server
3. Wait until TBSM completes the server synchronization.
4. Start the dashboard servers.

3.7 Installation

3.7.1 Installing the Fix Pack

[Fix Pack] Refers to the directory where you extracted the Fix Pack files from the zip file.

Attention: Install the Fix Pack with the same user that was used to install TBSM 6.2.0 GA Base release

Follow this procedure to install the Fix Pack:

1. On the Tivoli Business Service Manager server host, change to the [Fix Pack]/TBSM directory where you extracted the files.

2. Run the installation using one the following commands:
```
update_console_<arch>.* (Update in console mode)
update_gui_<arch>.*     (Update in GUI mode)
update_silent_<arch>.*  (Update in Silent mode)
```

Installation command options:

GUI mode, for example:
```
Windows: update_gui_win.bat

  or

UNIX® : ./update_gui_<unixplatform>.sh
```

Console mode for example -
```
Windows: update_console_win.bat

  or

UNIX® : ./update_console_<unixplatform>.sh
```

Silent Mode

a. Make a copy of the response file:
  6.2.0-TIV-BSM-<ARCH>-FP0001/*/scripts/sampleResponseFiles/<ARCH>/*_resp_update_<arch>
  to a location it can be edited (/tmp).

b. Update the file to match your environment.
  - Update the "profile id" variable (if necessary!)
    (refer to the instructions within the response file)
  - Update the "installLocation" variable (if necessary!) to indicate the location where the
    TBSM component is installed.

c. Run:

```
UNIX® : update_silent_<arc>.sh /tmp/Dash_resp_update_<arch>.xml -acceptLicense
               (or Data_resp_update_<arch>.xml )
```

or

Windows: i) Right-click the Windows Command Line icon
  ii) Click Run as Administrator
  iii) In the Windows command prompt, enter the following
    command to start the silent update:

```
update_silent_win.bat c:\tmp\Dash_resp_update_win.xml -acceptLicense
                   (or Data_resp_update_win.xml )
```

Note: The response file needs to have a fully qualified path.

3. After each component has been updated, restart the processes/services
   associated with those components!
   - The TBSM Data server (Impact Data and Dashboard)
   - The JazzSM process (where the TBSM Dashboard runs)

4. After installing this Fix Pack, end-users may need to clear their
   browser cache, close and re-open the browser; in order to avoid
   any issues on the client side.

3.7.2 Discovery Library Toolkit

NOTE 1: TADDM 7.2.0 and 7.2.1 are no longer supported.
  Refer to the following link for more info:http://ibm.biz
/Announcement_of_future_withdrawal_from_support_for_TADDM_7_2_0_and_7_2_1

3.8 TBSM Agent

Installing the TBSM Agent:

There are no updates to the TBSM Agent, with this Fix Pack (1).
IBM Tivoli Business Service Manager 6.2.0 Agent, can be downloaded from
the following location:

https://www-prd-trops.events.ibm.com/node/download-ibm-tivoli-business-service-manager-620-agent

3.9 Uninstalling the Fix Pack

Before uninstalling the FixPack all TBSM processes should be STOPPED. The user must manually stop the TBSM processes, before the uninstall of the FixPack (except for OMNIbus and DB2).

Attention: Uninstall the Fix Pack with the same user that was used to install TBSM 6.2.0.0 GA Base.

You can remove the Fix Pack by doing a rollback to a previous version, in this case, back to the TBSM 6.2.0. GA Base release.

This rollback process is different from uninstall, which removes the TBSM product entirely!

The rollback process can be completed by launching the Installation Manager, either the GUI mode (via IBMIM) or the command line interface (via imcl)

After the rollback process has completed, restart al of the TBSM associated processes and services.

==

4.0 Additional information for TBSM Fix Pack 1 APARS:

The following APARs fixed in this service delivery, require additional action by users. If your environment uses or is experiencing the symptoms described, use the instructions provided to complete the fix.

==

5.0 List of APARs

5.1 TBSM 6.2.0 Fix Pack 1 APARs

The following TBSM APARs are delivered with Fix Pack 1:

IJ11738: INCOMPLETE PAYLOAD FOR NODECLICKEDON EVENT WITH TBSM TOPOLOGY DATASOURCE
IJ12231: TBSM DB2JCC-9.7GA.JAR HAS KNOWN SECURITY ISSUES
IJ12459: DATA FETCH DAILY INTERVAL NOT AVAILABLE AFTER MIGRATION
IJ13129: NOT ABLE TO SAVE EXISTING DATA SOURCE ON TBSM 6.2
IJ13369: COLUMN NAME IN SERVICE INSTANCE DASHBOARD NOT GETTING WRAPPED IN TBSM 6.2
IJ14470: TBSM EXPORT Does Not set text=true for Numerical Formula Text Rules

List of TBSM APARs that have updates to the documentation:

IJ11636: TBSM 6.2 DOCS HAVE TIP DETAILS FROM VERSION 6.1.1
    Updated the section "Creating a freeform custom page" on the Service Configuration Guide and removed the Energy Dashboard Page information from the Customization Guide

IJ12534: UNABLE TO LAUNCH POLICY EDITOR FROM TBSM GUI
    Added a new topic "Enabling the Policy Editor from the TBSM Rules page" to the "Post Installation" section of the Installation Guide

5.2 Jazz for Service Management (JazzSM)

If JazzSM UI Services (DASH) is being used to visualize TBSM service model data, then the latest JazzSM fix pack should be installed after installing this TBSM FixPack. See Fix Central for availability:

http://www.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm~Tivoli&product=ibm/Tivoli/Jazz+for+Service+Management&release=All&platform=All&function=all

==

6.0 New support and features

6.1 TBSM 6.2.0 Fixpack 1

Fix Pack 1 includes the following enhancements:

None

6.2 Additional Browser Support with Fix Pack 1

Firefox ESR 60 (Quantum)
Internet Explorer 11

6.3 Support for the following Operating Systems is now available

----

RHEL 7.5
SUSE 12.3

6.4 Support for the following components is available

DB2 Version 11.1.2.2
JazzSM 1.1.3 Fixpack3 (DASH 3.1.3 FP3)
IBM Tivoli Netcool/OMNIbus 8.1.0.15
IBM Tivoli Netcool/OMNIbus Web GUI 8.1.0.15

==

7.0 Known issues -

For a complete list of Known Limitations, please refer to the following link:

http://ibm.biz/TBSMTroubleshooting

==

8.0 Copyright and trademark information

http://www.ibm.com/legal/copytrade.shtml

8.1 Notices

Please read THE IBM Tivoli Business Service Manager NOTICES AND INFORMATION in the file notices1.txt included in this Fix Pack before you proceed with the download and installation of this Fix Pack. You can find the additional notices file on the same web page as this Fix Pack.

8.2 THIRD-PARTY LICENSE TERMS AND CONDITIONS, NOTICES AND INFORMATION

w3-connections.ibm.com

Best Practices (TBSM) - Tivoli Netcool/Impact Wiki

2-3 minutes

Parameters

See the following explanation of the parameters:

cellName: The cell name of the dashboard server. The default value is TIPCell.

nodeName: Node name of the dashboard server. The default value is TIPNode.

serverName: The default value is server1.

datasourceName: Name of the data source used for the cluster.

datasourceJNDIName: The JNDI name of the data source. For example, jdbc/tipds. Use the same value of the DBDatasource property in the ha.properties file that is mentioned in step 2 for this setting.

datasourceHelperClassName: Use com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper for DB2.

providerName: Use TIP_Universal_JDBC_Driver _*as the _providerName*.

dsImplementationClassName: For DB2, use com.ibm.db2.jcc.DB2ConnectionPoolDataSource.

driverVarName: Use TIP JDBC_Driver_PATH _*as the _driverVarName*. __

driverClasspath: The root path where the DB2 JDBC driver is located. For this path, use {*TIP_root*}/universalDriver/lib. The property value of DBRoot in the ha.properties file should be the same as this one. On Windows systems, the typical value is C:/IBM/tivoli/tip/universalDriver/lib, while on Linux and UNIX systems, the typical value is /opt/IBM/tivoli/tip/universalDriver/lib.

user: The DB user name that is used to access the DB2 server.

password: The DB password that is used to access the DB2 server.

aliasName: The identifier for the user/password pair that is used by the data source to access the DB2 server.

aliasNameDesc: Description of the intended use of the user/password pair.

databaseName: The name of the database that is used for the dashboard server.

dbServerName: The host name of the new DB2 server.

dbPortNumber: The DB2 port that is used to accept incoming connection requests. The default value for DB2 is 50000.

dbDriverType: 4 for DB2 V9.x.

dbType: DB2 for the DB2 server.

Examples:

On Windows systems:

Assuming the dashboard server is installed in C:\IBM\tivoli\tip, run following commands to update the data source settings:

w3-connections.ibm.com

Advanced Topics - Tivoli Netcool/Impact Wiki

18-22 minutes

Advanced Topics

This section covers a variety of advanced **Tivoli Business Service Manager** topics.

TBSM 6.1.1 Topics Legacy Topics General Topics Planned Topics

TBSM 6.1.1 Topics

What's New?

These resources explain what's new in the latest TBSM deliverables available from Fix Central.

Back to topics

The "Integration" Series

These resources explain how to integrate TBSM service model data in JazzSM's Dashboard Application Services Hub (DASH).

An introduction to visualizing your TBSM service model data in DASH This document explains how TBSM has exposed its service model data via UI data providers. Topics are directly correlated to the appropriate TBSM product area.	Access this document **Updated:** 5/2/2016 [v2.2]
DASH Widget Recommendations for TBSM Datasets A handy table that provides guidance on which DASH widgets are appropriate for visualizing each TBSM dataset.	Access this document **Added:** 5/2/2016
Creating a high-level DASH-based dashboard to visualize your TBSM service model data [eDayTrader] This document explains how to create a high-level dashboard based on the eDayTrader sample application.	Access this document

This companion video demonstrates the actual building of the dashboard described in the above document. This video was created with the JazzSM 1.1.0 version of DASH. While the dashboard creation process remains the same, some details such as UI element names and/or capabilities have changed in subsequent fix packs.	Watch this video **Restored:** 8/20/2014
Sample TBSM Service Model: **eDayTrader** This artifact provides all the data and tools to install the sample eDayTrader service model. A document will be tagged with [eDayTrader] if this sample service model is relevant.	Access this artifact **Added:** 1/24/2014
Creating a drilldown dashboard to visualize support data [eDayTrader] This document extends the eDayTrader dashboard with a *drilldown* to another page. The data on the drilldown page is accessed from an external database using an Impact data type.	Access this document
Dataset Context Menus This document explains how to provide custom context menus on the complex DASH widgets (list, table, tree table, and topology) when visualizing TBSM service model data.	Access this document **Updated:** 5/2/2016 [v1.3]
Dashboard Best Practices This document presents some best practices that have evolved while developing various DASH-based dashboards especially when visualizing TBSM service model data	Access this document **Updated:** 7/6/2015 [v1.3]
Advanced Widget Usage This document provides examples of advanced DASH widget techniques which can be used in creating more advanced, elaborate dashboards or to meet specific business needs.	Access this document
TBSM Data + WebGUI Widgets This document explains how to integrate TBSM data and the widgets provided by OMNIbus WebGUI 8.1 in a DASH environment	Access this document **Updated:** 6/19/2015 [v1.2]
TBSM: How To Get Event Viewer on WebGUI 8.1.0 For AEL Replacement With the deprecation of browser plugins and the AEL on TBSM TIP, this document describes how a user can get the replacement for AEL, which is the Event Viewer on WebGUI 8.1.0	Access this document **Added:** 7/12/2016
Related Topic Out of the box, WebGUI's **Event Viewer** does not automatically publish **NodeClickedOn** events despite what is configured on the widget's **Events** panel.	**Added:** 9/9/2015

Additional widget configuration is required: you must set the widget's click action to "Update Event List (using wires)" to enable event publishing. Click here to access the WebGUI's Knowledge Center documentation	
TBSM: Navigating between TIP and DASH pages. This document provides an introduction to implementing navigation between TBSM dashboard pages, including AEL. Dashboards running on a TIP based TBSM Dashboard Server are covered as well as dashboards running on an IBM Dashboard Application Services Hub (DASH) server.**Note**: customer must setup "single signon" for smooth launching between TIP and DASHSee also section below **Bonus Navigation Senario**. Click here to access the WebGUI's Knowledge Center documentation	Dashboards-TBSM-Navigation_v1.3.pdf

Back to topics

The "Solution" Series

These resources explain how to implement various TBSM oriented solutions.

External Navigation between TBSM Pages on TIP and DASH This document explains how to implement external page navigation between TBSM dashboard pages, whether running on a TIP based TBSM Dashboard Server or a DASH server. The ability to extend the context menus of the advanced DASH widget (table, topology, or tree table widgets) is introduced for the first time.	Access this document **Updated:** 7/29/2014 [v1.3] Access document's artifacts **Updated:** 7/29/2014 [v1.3]
Related Topic "XLaunch Support in Dashboarding widgets" - explains how to externally launch a DASH page	Click to open this topic **Added:** 2/20/2014
Bonus Navigation Scenarios This document describes additional navigation scenarios with other products, like WebGui's Active Event List (AEL).	Access this document **Updated: 7**/6/2015 [v1.2]
Advanced "SmartText" Widget Usage This document provides advanced usage examples of the **Smart Text** widget that was shipped with JazzSM 1.1.1. **v1.2** - covers use of SVG-based symbols; now includes artifacts	Access this document **Updated: 3**/19/2015 [v1.2] Access document's artifacts **Added:** 3/19/2015

	[v1.2]
Visualizing TBSM Metric History Data This document describes how to access TBSM metric history data for visualization on a DASH-based dashboard. This is the same data visualized by TBSM's Time Window Analyzer. Because the TBSM UI data provider does not currently provide a dataset for this purpose, Impact capabilities are used to provide access to this data.	Access this document **Added:** 2/6/2014 Access document's artifacts **Added:** 2/6/2014
Sample Solution to Use Metric History data with DASH Line Chart [builds on "Metric History Data"] Thanks to Sergey Kolomiets and Mark Barinstein for a sample solution that scopes the metric history to a smaller interval and ensures a consistent set of data points for the interval. This provides data well suited for use by the DASH "Line Chart".	Access this document **Added:** 9/23/2014 Access document's artifacts **Added:** 9/23/2014
TBSM Event Transformation This document explains (and provides working samples) how the standard service context published by TBSM da- tasets in **NodeClickedOn** events can be transformed to enable integration with other products such as WebGUI and ITM.	Access this document **Added:** 3/16/2015 Access document's artifacts **Added:** 3/16/2015

Back to topics

TBSM UI Data Provider

These resources provide more information about the TBSM UI Data Provider.

Datasets	**Since**	**History**	
All Services	6.1.1.0	**Updated:** 5/2/2016	Access this document
Business Impact	6.1.1.4	**Added:** 5/2/2016	Access this document
Map	6.1.1.4	**Added:** 5/2/2016	Access this document
Templates	6.1.1.0	**Updated:** 5/2/2016	Access this document
Topology	6.1.1.0	**Updated:** 5/2/2016	Access this document
Tree Template [eDayTrader]	6.1.1.1	**Updated:** 5/2/2016	Access this document
		Updated: 5/2/2016	Access document's artifacts
Urgent Services	6.1.1.3	**Updated:** 5/2/2016	Access this document

.

TBSM UI Data Provider This document provides an introduction to the TBSM data provider.	Access this document
Managing Template Datasets This document describes how to manage which Template-based datasets are available for use via DASH widgets.	Access this document
Data Integration This document provides operational details about the TBSM datasets and how to integrate with them.	Access this document **Updated:** 5/2/2016 [v1.3]
TBSM Service Parameter for TBSM Datasets This document provides information about the effective use of the **TBSM Service** dataset parameter.	Access this document **Updated:** **5/2**/2016 [v1.5]
Dynamic Updating and the TBSM UI Data Provider This document explains how the TBSM UI Data Provider keeps its datasets updated, how this might affect widget usage in DASH, and how to configure the behavior of the dynamic updating.	Access this document **Updated:** **5/2**/2016 [v1.2]

Back to topics

OSLC Hover Preview

Deprecation for Jazz™ for Service Management Registry Services is due with version 1.1.3 and thus the Open Services Lifecycle Collaboration (OSLC) Hover Preview functionality, which uses this feature, is also deprecated, when used with versions or Jazz™ for Service Management equal to or greater than version 1.1.3.

These resources provide more information about TBSM's OSLC Hover Preview support.

Overview and Configuration Access the **Customization Guide**'s coverage of this topic.	Access this document

Back to topics

Legacy TBSM Topics

These topics are applicable to the legacy TBSM user interface (vs. DASH):

Topic	File and/or Document	Applies To
"Working with Custom Canvases" a document and other resources that cover advanced custom canvas activities like background images and extending the **SVG Image** widget.	Access this document	TBSM 6.1 until FP4

Topic	File and/or Document	Applies To
The Java plug-in for web browsers relies on the cross platform plugin architecture NPAPI, which has been supported by all major web browsers for over a decade. However, Mozilla intends to remove support for most NPAPI plugins in Firefox by the end of 2016. Oracle has announced that it is to depreciate the Java browser plug-in technology with the next release of the Java platform (JDK9). Because of these industry moves, TBSM is announcing the end of support for the Service Viewer (and custom canvases). These features will be removed from the product in Fix Pack 5. Customers are advised to replace custom canvas usage with IBM Dashboard Application Services Hub pages, as soon as possible.		TBSM 6.1.1 until FP4
"Disabling Integration with ITCAM for Internet Service Monitors" TBSM 6.1.1 Fixpack 4 has disabled the integration with ITCAM for Internet Service Monitors (ISM) due to architectural changes in more recent versions of the ISM product. This document describes the impact to the TBSM administrator and operator of disabling this integration. For users that may have ISM artifacts in there TBSM system, like the ISM servers and ISM rules, this document describes utility functions that can be used to clean up these artifacts.	Access this document	TBSM 6.1.1 FP4

General Topics

- Business Service Composer

Planned Topics

- Resource Enrichment Techniques
- Discovery Library Toolkit Policy Rules
- Import/Export Scenarios

Business Service Composer documentation

]>

Creating Cognos Report.html

Loading your TBSM 6.1 Service Model into TBSM 6.1.x Milestone Environments

]>

The TBSM 6.1.x Milestone 1 driver only supports new installs. A future Milestone driver will enable upgrading an existing 6.1 FP1 system to the full 6.1.1 level which will enable full access to your existing service models for creating custom dashboards.

To get started creating TBSM custom dashboards using the TBSM 6.1.x Milestone 1 driver and TIP 3.1, it may be helpful to recreate all or part of your TBSM 6.1 service model in your TBSM 6.1.x Milestone 1 environment. Using TBSM radshell commands, you can export service instances and templates from your TBSM 6.1 Data server and import them into your TBSM 6.1.x Data server. For more information about using radshell commands, see the TBSM 6.1 Administrator's Guide.

The following steps export and import your TBSM 6.1 service model.

Export your service model from your TBSM 6.1 Data server

1. On your TBSM 6.1 Data server, from a command prompt, type the following command and press Enter:

UNIX	$TBSM_HOME/bin/rad_radshell
Windows	%TBSM_HOME%\bin\rad_radshell

2. When the radshell> prompt is displayed, type one of the following rad_shell commands and press Enter:

export();
Use the export function to export your all of your TBSM service model to a radshell file. When you use the export function, all of your service instances, templates, data fetchers and data sources are written to radshell file named export.radsh in the $TBSM_HOME/export directory.

exportFromStartingInstance("starting-instance-service-name");
Use the exportFromStartingInstance function, providing the starting instance service name, to export part of your TBSM service model to a radshell file. When you use the exportFromStartingInstance function, a service instance and its related templates, as well as all of the underlying child service instances and their related templates, are written to a radshell file named export.radsh in the $TBSM_HOME/export directory.

exportMeta();
Use the exportMeta function to export your TBSM service model's configuration to a radshell file. When you use the exportMeta function, all of your templates, data fetchers, data sources, and auto-population rules are written to a radshell file named exportMeta.radsh in the $TBSM_HOME/export directory. Service instances are not exported when using this function.

3. Type exit(); and press Enter.

Import your service model into your TBSM 6.1.x Data server

On your TBSM 6.1.x Data server, from a command prompt, type the following command and press Enter:

UNIX	cat *radshell-file-name* | $TBSM_HOME/bin/rad_radshell
Windows	type *radshell-file-name* | %TBSM_HOME\bin\rad_radshell

where *radshell-file-name* is the fully qualified name of the radshell file that contains the output of the export command you used.

---

regarding IBM's future direction or intent are subject to change or withdrawal without notice and represent goals and objectives only. Please refer to the developerWorks terms of use for more information.

TBSM UI Data Provider - Managing Template Datasets

By default, TBSM has UI data provider datasets for every template in the service model, including, the BSM templates installed by TBSM.   This results in a lot of datasets being listed by the Dashboard Application Services Hub (DASH) console Quick Edit function when configuring a widget. You can use the search capabilities to filter down the list, but it is possible to configure TBSM to show fewer datasets.

If there are templates in TBSM for which you will never need UI provider datasets, then you can configure TBSM to exclude these templates. The attached sample file illustrates commands that can be used with the TBSM rad_radshell utility to configure TBSM to exclude one or more templates from providing datasets. This sample shows how to exclude all of the BSM templates that are installed by TBSM, in case these are not templates that are assigned to services in your TBSM service model.

You can modify this sample file as needed to:

- exclude your own templates from providing datasets
- restore one or more templates so they resume providing datasets

See the comments in the attached file for detailed instructions.

This configuration can also be accomplished from the TBSM console, but if many templates are to be modified, then this command line approach will be more efficient.

Sample file for managing template datasets

Reference

TBSM and IBM Dashboard Application Services Hub

The TBSM data provider

Excluding a template as a dataset

Administering TBSM using the RAD shell tool

Notices

References in content to IBM products, software, programs, services or associated technologies do not imply that they will be available in all countries in which IBM operates. Content, including any plans contained in content, may change at any time at IBM's sole discretion, based on market opportunities or other factors, and is not intended to be a commitment to future content, including product or feature availability, in any way. Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice and represent goals and objectives only.

./TBSM UI Data Provider - _Top.html

Starting with **TBSM 6.1.1**, a UI data provider makes TBSM service model data available for visualizing in the IBM® Dashboard Application Services Hub console component of Jazz™ for Service Management. The TBSM UI data provider is accessed from DASH by defining a remote connection to a TBSM 6.1.1 or later Dashboard server.

Information is available in the TBSM **Knowledge Center**: http://www-01.ibm.com/support/knowledgecenter/SSSPFK/welcome

1. open the documentation for the appropriate TBSM release; for example, 6.1.1

2. *open the **Administrator's Guide***
3. *open the **TBSM and IBM Dashboard Application Services Hub** topic*

The items contained in this section will provide information on advanced topics related to the TBSM UI data provider.

Notices

w3-connections.ibm.com

Integration Scenarios - Tivoli Netcool/Impact Wiki

225-286 minutes

This section of the wiki includes Tivoli Business Service Manager integration scenarios. The following integration scenarios are available:

Best Practices Integration of IBM Tivoli Monitoring and IBM Tivoli Business Service Manager for Databases - A Dashboard for Database Administrators

Product integration overview

This document presents a solution showing the integration of IBM® Tivoli® Monitoring with IBM Tivoli Business Service Manager (TBSM) within the context of database administration. For more information on business service management, see http://www.ibm.com/software/tivoli/solutions/bsm.

You use Business Service Management to connect IT with the business aspect of your enterprise. The Business Service Management approach forces IT personnel to think in terms of the business context and how the availability and performance of their IT systems affects the enterprise.

This document describes a detailed, step-by-step solution showing vital KPI (key Performance Indicators) for databases such as DB2, MSSQL, and Oracle.

KPIs can be classified into three types:

- Management, which includes the achievement of business goals on schedule
- Financial, which includes resources such as cost per project, data centers, energy, and other financial goals
- Operations, which includes monitoring the percentage of CPU, energy, and other operational resources that are used in a given data center

Simply put, KPIs help businesses understand how well an organization is accomplishing their objectives and goals.

The key products used in this solution are IBM Tivoli Monitoring (http://www.ibm.com/software/tivoli/products/monitor) and IBM Tivoli Business Service Manager (http://www.ibm.com/software/tivoli/products/bus-srv-mgr).

IBM Tivoli Monitoring **(ITM)** is an enterprise-class solution that optimizes IT infrastructure performance and availability. Tivoli Monitoring software manages your IT infrastructure, including operating systems, databases and servers across distributed and host environments through a single customizable workspace portal.

IBM Tivoli Business Service Manager **(TBSM)** provides enterprise and service provider organizations with advanced service and process visibility in targeted, real-time dashboards.

IBM Tivoli Monitoring setup

See the following information for setting up Tivoli Monitoring:

System Details:

sysitm.tivlab.austin.ibm.com

Windows 2003 SP2 Enterprise Edition

2 CPUs, 1 Gigabyte Memory

Here are details of the IBM Tivoli Monitoring Components Version:

Installed under C:__IBM\ITM directory:

Managed System Name Product Code Version Status

Primary:CHAFE:NT NT 06.20.00.00 Y

Primary:ARCHITECT:NT NT 06.20.01.01 Y

Primary:SYSITM:NT NT 06.20.00.00 Y

sysitmASFSdp:UAGENT00 UA 06.00.00.00 Y

systbsm:LZ LZ 06.20.00.00 Y

ducati:LZ LZ 06.21.00.00 Y

SYSITM:UA UM 06.20.00.00 Y

WAREHOUS:CHAFE:ORA OR 06.20.00.00 Y

ARCHITECT:ARCHITECT:MSS OQ 06.20.00.00 Y

MSDE_VC:ARCHITECT:MSS OQ 06.20.00.00 Y

db2inst1: Ducati:UD UD 06.20.00.00 Y

db2inst2: Ducati:UD UD 06.20.00.00 Y

DB2:SYSITM:UD UD 06.20.00.00 Y

systbsm:R9 R9 04.11.00.00 Y

SYSITM:Warehouse HD 06.20.00.00 Y

HUB_SYSITM EM 06.20.00.00 Y

Details can be gathered from the installation and user's guides for IBM Tivoli Monitoring at http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?toc=/com.ibm.itm.doc/toc.xml, and for IBM Tivoli Monitoring for Databases at http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?toc=/com.ibm.itmfd.doc/toc.xml.

TBSM setup

See the following details for TBSM setup:

System details:

systbsm.tivlab.austin.ibm.com

Red Hat Enterprise Linux AS release 4 (Nahant Update 6)

2 CPU, 1 Gig Memory

Here are the TBSM Components Version details:

Installed in the /opt/IBM/Netcool directory:

Tivoli Business Service Manager (TBSM) Version - *4.1.1*

4.1.1.0-TIV-BSM-IF0001

4.1.1.0-TIV-BSM-IF0001 (NGF 1.1.392)

4.1.1.0-TIV-BSM-IF0006

License Server - 1.031

Omnibus - 7.1

Security Manager - 1.4

Discovery Library Toolkit - 4.1.1

EIF (Event Integration Facility) Probe

Note: Use the Discovery Library Toolkit included with TBSM v4.2 or ensure that the 4.1.1.0-TIV-BSM-IF-0007 interim fix is installed_._

Loading data from IBM Tivoli Monitoring into TBSM

In the preceding section under IBM Tivoli Monitoring setup, the Tivoli Monitoring environment has all database and OS agents configured and connected. This information is exported from Tivoli Monitoring by using the Tivoli Monitoring Services Discovery Library adapter.

Tivoli Monitoring Services Discovery Library adapter overview

The Tivoli Monitoring Services DLA is an executable program that is included with Tivoli Monitoring. The Tivoli Monitoring Services Discovery Library adapter (DLA) scans the Tivoli Monitoring environment and identifies the managed systems in the environment. You can then feed this information (using an XML output file) into IBM Tivoli Change and Configuration Management Database or another CMDB. The DLA identifies all distributed and mainframe agents, and logical groupings defined through the Tivoli Enterprise Portal.

The DLA gathers information by querying the hub monitoring server for all managed systems and mapping them to Common Data Model resources based on the agent product code and managed system name format. The monitoring servers and portal server must be running to use these queries. The managed systems can be running or not running. The DLA is run from the command line on the computer where the portal server is installed. The command is located in the \IBM\ITM\CNPS subdirectory.

By default, the DLA generates the XML output file in the \TMSDLA subdirectory on the portal server. The name of this file follows the standard Discovery Library file name format, such as TMSDISC100.systemA.ibm.com.2006-05-22T23.35.06Z.refresh.xml. To use this information in the CMDB, you must transfer the XML file to the Discovery Library File Store and then use the Discovery Library Bulk Loader.

In addition to discovering resources and relationships, the Tivoli Monitoring Services DLA discovers information that the IBM Tivoli Change and Configuration Management Database uses to provide a context sensitive launch to the Tivoli Enterprise Portal. You can also view the status of the discovered managed systems while in IBM Tivoli Change and Configuration Management Database.

Extracting the Discovery Library book from IBM Tivoli Monitoring

This section describes the steps to extract the Discovery Library book from the IBM Tivoli Monitoring system using the KfwTmsDla.exe command.

Note: The KfwTmsDla.exe command is located in the Tivoli Enterprise Portal Server (or portal server) component of Tivoli Monitoring. If you have a portal server and Tivoli Enterprise Monitoring Server (or monitoring server) on separate computers, the command is located on the portal server computer.

Run the KfwTmsDla.exe command as follows:

C:\IBM\ITM\CNPS>KfwTmsDla.exe ?

TMS Discovery Library Adapter Command Line Utility.

Usage: KfwTmsDla [/?][/b][/d][/l][/o orgname][/s][/u][/x outputfile]

Options:

/? Display this help screen.

/b Launch html browse utility on completion of the discovery.

/d Create diagnostic file during discovery process.

/l Discover Logical Views.

/o orgname Organization GlobalName will be set to orgname.

/s Discover HTTPS URL rather than HTTP URL.

/u Map to Unitary ComputerSystem CDM classes.

/x outputfile Create XML file of name outputfile.

C:\IBM \ITM\CNPS>KfwTmsDla.exe

Initializing TMS Discovery Library Adapter

Discovering Managed System Agents.....17 agents discovered

Discovering Hub TEMS...

Writing out Managed System Agents....Complete

23 Agents written out

3 Agents suppressed

5 Operating Systems written out

0 Sysplexes written out

TMS Discovery Complete. Results in C:\IBM\ITM\cnps\tmsdla\ TMSDISC100-A.sysitm.tivlab.austin.ibm.com.2008-08-05T20.40.49Z.refresh.xml

Loading the Discovery Library book into TBSM

The Discovery Library book created in section 3.1 is copied to the TBSM system under the */opt/IBM/Netcool /discovery/dlbooks* directory.

The TBSM XML Toolkit Service is installed under the */opt/IBM/Netcool/XMLtoolkit/bin* directory and includes the xmltoolkitsvc.properties properties file_._

The following two parameters and others listed in the xmltoolkitsvc.properties file determine how often the XML Toolkit Service looks for a file under the specified location:

DL_PollIntervalSeconds = 15

DL_FileSystem = /opt/IBM/Netcool/discovery/dlbooks

When the file is copied to the *msgGTM_XT.log* file under */opt/IBM/Netcool/XMLtoolkit/log*, you can verify that the DLA book was successfully loaded into TBSM:

GTMCL5290I: Book TMSDISC100-A.sysitm.tivlab.austin.ibm.com.2008-08-05T20.40.49Z.refresh.xml processed successfully.

Verifying Discovery Library book import

When the Discovery Library ***book is loaded, log on to the TBSM console (_http://systbsm:8080_) and click *Service Administration**.

Drill down into the navigation tree to see all the Tivoli Monitoring OS agents and DB Agents listed under their respective categories.

Note: The SCR (Service Component Registry) templates are loaded when you select the **simple** option during installation. If you select the **advanced** option, you must load the templates by hand as shown previously.

If you cannot see the SCR templates displayed in the TBSM console after the installation is completed, you must manually load the templates. Download the BSM_Templates.radsh file into TBSM as follows:

Cat /opt/IBM/Netcool/guifoundation/webapps/sla/install/BSM_Templates.radsh | /opt/IBM/Netcool/bin/rad_radshell

Note: If the Discovery Library Toolkit included with TBSM v4.2 is used, the deficiency listed below can be avoided. Otherwise, you must manually perform the following i) and ii) steps.

To view the Oracle Server in the Oracle category, use the following steps:

1. Under the **Service Component Registry** tab, expand **Component Registry**.
2. Expand **Application Servers**.
3. Expand **Databases**.
4. Click **Oracle**.
5. Under the Service Viewer, click **Edit Service SCR_OracleDatabases**.
   The value for classname filter currently contains the following value:

   'cdm:app.db.oracle.OracleDatabase','cdm:app.db.oracle.OracleInstance'
6. Change this classname filter to the following value:
   'cdm:app.db.oracle.OracleDatabase','cdm:app.db.oracle.OracleInstance','cdm:app.db.oracle.OracleServer'

   The Oracle Server is displayed in the component registry tab.

Another deficiency you can encounter is the BSM _Identity for the OS Agents, which is displayed as the following value:

BSM_Identity = sysitm.tivlab.austin.ibm.com(Signature=9.48.139.114)-ComputerSystem

Replace the ITM OS Agent Name with the following value:

BSM_Identity = Primary:SYSITM:NT

Creating Dashboards in IBM Tivoli Business Service Manager (TBSM)

In previous steps, you took actions to import the data from Tivoli Monitoring to TBSM. Because all the OS Agents and Database Agents are now listed under the Component Registry, the next step is to create Templates, Rules, Services, and Dashboards.

Note: It is very important that serious consideration is given when you are creating the preceding categories to minimize the efforts. A design that is carefully reviewed will save time and will produce a much accurate picture at the dashboard level.

Now you are going to create the following two dashboards: the IT Operations View (IT Operations Centric Dashboard and the DBA View (DBA Operations Centric Dashboard).

- IT Operations View (IT Operations Centric Dashboard)

This dashboard consists of a Service Tree, Service Viewer, Service Details, and IFrame Example (Used for Custom Canvas).

- DBA View (DBA Operations Centric Dashboard)

This dashboard consists of a Service Tree, Service Viewer, and two IFrame Examples (Used for Custom Canvas and Webtop).

Use the following generic steps to build a dashboard:

1. Create queries in the portal server (with appropriate agents assigned to that query).
2. Get the query ID from the portal server database.
3. Lay down the code on the TBSM system to extract data from the Tivoli Monitoring system.
4. Create policies (based on data fetchers) to extract data from the Tivoli Monitoring system (using the query ID and calling the code laid down to extract data).
5. Assign the data fetcher-based policy to a template.
6. Assign a template to a service.
7. Assign a template to the appropriate column in a dashboard's service tree.
8. Create the tree template.

Building the IT Operations View Dashboard

The IT Operations view Dashboard has the following four views:

- Service Tree
- Service Viewer
- Service Details
- IFrame Example (Used for Custom Canvas)

Service Tree View

For the Service Tree view, the **State, Event Status, Real-time CPU Usage** and **Real-time Memory Usage** columns in a Service Tree are displayed:

You will create templates using data fetchers based on the policy to extract real-time data from IBM Tivoli Monitoring.

Create Queries via portal client in IBM Tivoli Monitoring

Start with IBM Tivoli Monitoring. As in the preceding dashboard requirement, you need to collect %CPU and %MEMORY used metrics in real-time from Tivoli Monitoring. You need to create queries using the Tivoli Enterprise Portal Client (or portal client) in the portal server, which will gather that information. Then you will extract that data using the IDs of the queries in a Java® program and bring the data into the TBSM dashboard using a policy data fetcher.

The workspace is created in the portal server using the portal client. This workspace has four views, targeting the CPU/Memory metrics for Linux and Windows systems.

The query in the query editor retrieves the *CPU data for Windows System* from the NT_Processor Attribute group using the Server Name, Processor, % Processor Time, and Timestamp metrics.

Note: The order of the metrics listed in this query is very important because that is the order they will be fetched by the policy in TBSM.

The query in the query editor retrieves the *Memory data for Windows System* from the NT_Memory Attribute group using the Server Name, % Committed Bytes In Use, and Timestamp metrics.

Note: The order of the metrics listed in the query is very important because that is the order they will be fetched by the policy in TBSM.

The query in the query editor retrieves the *CPU data for Linux System* from the Linux_CPU Attribute group using the System Name, CPU ID, % System CPU and Timestamp metrics.

Note: The order of these metrics listed in the query is very important because that is the order they will be fetched by the policy in TBSM.

The query in the query editor retrieves the *Memory data for Linux System* from the Linux_VM_Stats Attribute group using the System Name, % Real Memory Used, and Timestamp metrics.

Note: The order of these metrics listed in the query is very important because that is the order they will be fetched by the policy in TBSM.

You will now log on to the portal server database to retrieve the ID for the queries created previously.

Laying the code (or jar files) on IBM Tivoli Business Service Manager (TBSM) to extract real-time data from IBM Tivoli Monitoring

You must copy the following jar files to the TBSM system:

- ITMaccess.jar
- ITMcnp_vbjorball.jar
- ITMkjrall.jar
- ITMutil.jar
- ITMbrowser.jar
- ITMice.jar
- ITMtap_cli.jar

Copy the jar files into the

/opt/IBM/Netcool/guifoundation/webapps/sla/WEB-INF/lib directory before you create the policies to extract data under the */opt/IBM/Netcool/guifoundation/webapps/sla/policy* directory.

Enabling Tivoli Monitoring Data Access in TBSM

Add the following four lines at the end of the RAD_actionlist file located under the /opt/IBM/Netcool/etc/rad/ directory:

- impact.actions.numactions=40
- impact.actions.40.name=ItmDataAccess
- impact.actions.40.isbuiltin=true
- impact.actions.40.class=com.micromuse.sla.impact.ItmDataAccess

Note: If you already have 39 actions in the RAD_actionlist file, then your next number is 40, as shown previously. And you must restart TBSM services.

Create policies in IBM Tivoli Business Service Manager (TBSM) for each of the preceding queries.

The policies that you create will extract data from Tivoli Monitoring into the Tivoli Enterprise Portal Server using APIs.

In the following policies that are created, the key is this line in the policy file:

ItmDataAccess("win_agents", "UDQ-108.10.14-17.49.15-00006", "sysitm.tivlab.austin.ibm.com", "sysadmin", "password");

- The first argument is the SytemName or MSL name from Tivoli Monitoring.
- The second argument is the Query ID of the custom query created in Tivoli Monitoring.
- The third argument is the host name where the portal server is located.
- The fourth argument is the Username.
- The fifth argument is the Password.

Note: You must update the preceding fields according to your environment.

Policy file to extract real-time data from the IBM Tivoli Monitoring for Windows CPU metric:

Note: The myRow object is mapped to the metrics in the same order because the metrics were defined in the query in the portal server.

Go to */opt/IBM/Netcool/bin* and run the **rad_radshell** script that creates the policy:

Policy file to extract real-time data from Tivoli Monitoring for Windows Memory metric:

Note: The *myRow* object is mapped to the metrics in the same order because the metrics were defined in the query in the portal server.

Go to */opt/IBM/Netcool/bin* to run the **rad_radshell** script that creates the policy:

Policy file to extract real-time data from the IBM Tivoli Monitoring (ITM) for Linux CPU metric:

Note: The *myRow* object is mapped to the metrics in the same order because the metrics were defined in the query in the portal server.

Go to */opt/IBM/Netcool/bin* to run the **rad_radshell** script that creates the policy:

Policy file to extract real-time data from the IBM Tivoli Monitoring for Linux Memory metric:

Note: As you will note, the *myRow* object is mapped to metrics in the same order as the metrics were defined in the query in the portal server.

Go to */opt/IBM/Netcool/bin* to run the *rad_radshell* script to create the preceding policy.

Verifying Data Fetchers in IBM Tivoli Business Service Manager (TBSM) Console

Use the following procedure to verify that all the data fetchers created by the command line are displayed in the TBSM Console:

1. Log on to the TBSM Console.
2. From the drop-down menu, select the Service Administration page.
3. In the Service Navigation, click the **Data Fetcher** tab. All four of the previously created policies are displayed.

Create Templates & Rules

Now you will create the templates and rules for the previously created policies.

You created the following two templates because the OS agents are running on Linux and Windows OS:

- GetLinuxRealTimeData
- GetWinRealTimeData

Under the GetLinuxRealTimeData template, you create the two Incoming Status Rules (for CPU and Memory metrics) based on Numeric Value as shown in the following screen capture:

Numeric Incoming Status Rule for Linux CPU Metric:

Numeric Incoming Status Rule for Linux Memory Metric:

Looking under the GetWinRealTimeData, you create the Incoming Status Rule (for CPU and Memory metrics) based on the Numeric Value that is shown in the following screen capture. These rules will retrieve the real-time data via the policies from Tivoli Monitoring.

Numeric Incoming Status Rule for Windows Processor Metric:

Numeric Incoming Status Rule for Windows Memory Metric:

Create the Service structure

Create the Service structure, which will list all the Physical Systems (OS Agents):

The templates are assigned to these systems (as they were loaded via the DLA book from Tivoli Monitoring). The only manually added template is the GetWinRealTimeData template because it retrieves real-time data via the policy from Tivoli Monitoring.

Note: Do not change the rules in these SCR service templates (SCR_*). These templates are pre-configured to create and monitor services from objects in the IBM Common Data Model. If you change these service templates, you can disable this feature. If you want to create additional rules for a service, create a new service template, configure the rules you want, and assign your service to the new service template.

Here is a screen capture of one of the systems named architect.tivlab.austin.ibm.com:

Assigning the Values to ServerName under the Service, manually

The **BSM_Identity** field helps to map the events coming from Tivoli Monitoring.

The **ServerName** fields help to map the real-time data from Tivoli Monitoring.

Create the Tree Template

In this section you will create customized tree structure to be displayed in the service tree view.

Under the Service Navigation page, click the **Tree Template editor**.

Create a Tree Template named "IT Operations":

Create the fields for which you will assign data, as shown in the following screen capture. In this case, you create two columns named "% RealTime CPU" and "% RealTime Mem":

Now assign each Available Attributes under each Active template with a desired column target.

Create the Custom Canvas View

The Custom Canvas for US map is created under the Service Administration for DataCenters Service.

Click the **DataCenters** service. In the service viewer you will see a graphical layout in the *view service 'DataCenters' tab.

Click the **Create Custom Canvas** icon and **Decoration** tab. Click **IBM image** (shown as **Basic Image**) to add your own custom image.

The window opens when you select the **BasicImage icon IBM**, and the iconLocation and displayName is replaced by the path of the desired image file:

/opt/IBM/Netcool/guifoundation/webapps/sla/images/usaa_map.PNG

Click **View > Inspector** to view the properties of the image:

To assign the icon on each state, the Indicator, ServiceInstance icon is used and a service instance name is attached to each state.

The Configure New Instance window opens. Select the **CA_DataCenter** service name:

In the next window, select the layout of the icon.

Now you can capture the URL of this custom canvas created previously.

To retrieve the URL, see the bottom of the Internet Explorer window, as highlighted by the arrow markers.

Then you can take the Service Instance ID number (113 in this case) and use that in URL to provide it to IFrame, as shown here:

/sla/av/ServeViewerApplet?CanvasTemplate=CustomView&ServiceTypeID=1&ServiceInstanceID=113

Templates and Rules for Custom Canvas Services

Behind these services, templates and rules (Incoming Status Rule) are based on a Good, Marginal, and Bad Threshold. The AlertKey is set to match the Situation Names, which are created in Tivoli Monitoring using the Location naming convention.

The following example shows the Incoming status rule based on Good, Marginal, and Bad Threshold rule:

Create View under Desktop Pane

After you assign the templates and rules to the desired column names, you can create a view on the dashboard for IT Operations under the desktop pane.

Go to the */opt/IBM/Netcool/guifoundation/webapps/desktop/actionsDemo/view* directoryand make a copy of the

Service View_actions.xml file using the exact name you want to give the view in the desktop pane. In this case, call it IT Operations, or IT Operations View_actions.xml.

Go to the Desktop Pane and create a view called IT Operations View.

Note: Make sure the layout is set to the **State-Maintained Tab** pane.

Click **Save and apply**.

In the IT Operations View, the following viewpoints are added:

- Service Tree
- Service Viewer
- IT Data Centers (based on Custom Canvas)
- Service Details

In the IT Operations View, you have two custom viewpoints, Service Tree and IT Data Centers.

In the following screen capture, the Service Tree viewpoint is created by selecting the **Service Instance** you built under **Service Navigation > Services** tab.

The following screen capture shows that the IT Data Centers viewpoint is created by using the Custom Canvas, which shows the Service Instance you built under the **Service Navigation > Services** tab.

Note: For two Service Viewers to exist on one page and act independently, you must use IFrame for the second service viewer and use the URL (shown previously in section 4.1.2) to get to custom canvas as shown below.

Building the View Dashboar

The DBA view dashboard has the following four views:

- Service Tree
- Service Viewer
- IFrame Example (used for Custom Canvas)
- IFrame Example (used for Webtop view)

Service Tree View

For the Service Tree view the following columns are shown in a Service Tree: **State, Event Status, Bufferpools, Locks,** and **Tablespaces**. Icons are used to show the state of KPIs in the Service Tree, such as cloud, rain, and sunshine.

Images for sun, cloud, and thunder are placed in the /opt/IBM/Netcool/guifoundation/webapps/sla/images directory_._

For the icons to be displayed in the Service Tree view, the following file is used:

/opt/IBM/Netcool/guifoundation/webapps/sla/policy/RAD_GetTreeColumnValue.ipl

This piece of code is added to RAD_GetTreeColumnValue.ipl file, so when the event severity changes, an icon is displayed.

IFrame for Database Statistics

For the Database Statistics view, services are created.

Behind these services, there are templates and rules (Incoming Status Rule) based on a Good, Marginal, and Bad Threshold, AlertKey that are set to match the Situation Names, which are created in Tivoli Monitoring using the Location naming convention.

The template is created, and again the AlertKey is used to match the Situation Names, which are created in Tivoli Monitoring using locks naming convention.

A custom canvas is created as described in section 4.1.2.

IFrame for Database Events Summary (using webtop)

For Events By Database Type, Webtop is used to show a variety of options.

Under the **Webtop Admin > Maps** section, the icons are selected (event histogram):

Behind this histogram lie the Entities, which filter on events coming through NetCool/Omnibus® from Tivoli Monitoring as shown here:

Entity for DB2 Database Agents

Entity for MSSQL Database Agents

Entity for Oracle Database Agents

Note: All of these customizations are available in TBSM v4.2 version, too, but include a slightly different layout.

Solutions for Download

To access and download the files (scripts, rules, etc) required by this best practices integration between Tivoli Monitoring for Databases and TBSM, click here. This link directs you to the IBM Open Process Automation Library (OPAL).

References

- http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.itm.doc_6.1/itm_install314.htm
- http://ibm.biz/TADDMLink
- http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool_wt.doc/ag/xF11248784180104.html

Acknowledgements

This white paper was authored by Naeem Atlaf. A special thank you goes to the following individuals for their contributions to this solution:

Cesar Araujo, Jeff Ferla, Ahmed Kira, Douglas McClure, Karl McCormick, Tod Thorpe.

Copyright

BSM Stack Upgrade Report July 2011

As part of IBM's commitment to quality and continuous improvement, the Tivoli Project Bedrock testing team was established. The team consists of cross-product test representatives from IBM Software Group,Tivoli software, and IBM Systems and Technology Group hardware divisions. Project Bedrock enhances the way IBM tests by doing scenario-based testing on a solution stack comprised of multiple Tivoli products, middleware, operating systems, and hardware.

Project Bedrock was created to address scenario-based testing to ensure the maintenance levels are installed, configured, and tested together as a solution stack. You follow a well-documented path to move from one level of the solution stack to the next. The Bedrock project name was chosen to portray the end goal of delivering a solid foundation of products working together on which the solutions run. Read more about the product versions used in the stack, procedures followed to implement the stack, issues found, and tips and tricks to learn from.

Products included:

- IBM AIX®
- IBM Systems Director
- IBM Tivoli Directory Server
- IBM DB2® Universal Database™
- IBM Tivoli Monitoring
- IBM Tivoli Composite Application Manager for Diagnostics (ITCAM for Diagnostics)
- IBM Tivoli Composite Application Manager for Transactions (ITCAM for Transactions)
- IBM Tivoli Application Dependency Discovery Manager (TADDM)
- Tivoli Integrated Portal
- IBM Tivoli Netcool® OMNIbus
- IBM Tivoli Netcool Impact
- IBM Tivoli Netcool Web GUI
- IBM Tivoli Network Manager for IP Networks
- IBM Tivoli Common Reporting
- IBM Tivoli Business Service Manager (TBSM)

To read this white paper, click here.

Building a BSM for Storage Solution

This paper is a step-by-step guide to building a business service management solution by using the IBM® Tivoli® Storage Productivity Center to monitor your storage devices. The IBM Tivoli Storage Productivity Center suite of storage infrastructure management tools can help customers improve time-to-value, and reduce the complexity of managing their storage environments by centralizing, simplifying, and optimizing storage tasks associated with storage systems, storage networks, replication services, and capacity management. The suite of Tivoli software integrates together to give you a picture of the overall health of your business services, including your storage controllers, fiber channel switches, server systems, and more. You can use this solution to help you stay ahead on the status of your business applications by knowing critical information such as availability and status, and being able to proactively respond to that information. Tivoli software integrates together to discover and monitor your servers and storage devices, collect related elements into a business application or service, and display notifications when events occur that effect the environment. From high-level visual overviews of your environment, you can drill down to more detailed information about the infrastructure and events happening within it.

In this document, we will perform the following steps:

1. Configure the Tivoli Productivity Center's rules file for Business Service Management events
2. Configure the TADDM's rules file for Business Service Management events
3. Configure TBSM for Tivoli Productivity Center's storage devices and events
4. Configure Tivoli Productivity Center to trigger alerts and send events
5. Performing a TADDM discovery of storage devices in Tivoli Productivity Center
6. Configure TADDM for change events
7. Bringing it all together in a TBSM dashboard

Current Limitations: The Tivoli Productivity Center sensor doesn't currently pull Virtualization relationships (i.e. VMWare), even though they are stored in the Tivoli Productivity Center database.

To read this paper, click here.

About the Author

This paper was written by Brian R. Fabec, an integration architect for Tivoli Service Availability and Performance Management solutions.

Business Service Management for SAP

Common TIP Container - Top

Usecase 1 - Installing IBM Tivoli Network Manager 3.9 in a TIP previously installed by Tivoli Business Service Manager 6.1

Common TIP Container - Usecase 1 - Installing IBM Tivoli Network Manager 3.9 in a TIP previously installed by Tivoli Business Service Manager 6.1

This document will guide you with the high level steps on how to install IBM Tivoli Network Manager IP Edition version 3.9 Refresh in a TIP that was installed by Tivoli Business Service manager. The main aim of this document is to highlight the special steps and considerations required for a successful installation of these two products in this order. Since this document only covers the high level and special steps the user will refer to the Install Guides of each product for the detail instructions.

Creating a Business Services Dashboard with IBM Tivoli Composite Application Manager for Transactions and IBM Tivoli Business Service Manager

This paper describes why your company needs to monitor your mission critical business services from the perspective of your user's experience, how you can use IBM Tivoli products to automatically create a complete end-to-end view of your business services, and how these tools simplify problem isolation and dramatically improve your business service availability.

Found here

Creating a TBSM Discovery Library Toolkit book based on an alternate namespace

Enriching ITM Events For TBSM

Business Services are a collection, grouping of Hardware, OS, Devices, Networks, Databases, Webservers, and Applications etc. presented on a dashboard for executives to make strategic and business decisions.

In a company, there can be thousand of systems running several different homes grown, 3rd party applications. To consolidate all the data coming from all these systems in form of events, from external resources such as databases (cmdb), they have to have some common data to identify and differentiate and map to a service being provided by the company. Basically tagging the data (events in this case), so they can be assigned to there appropriate services in the service tree to reflect an accurate picture of the services provided to end users.

In this document we will refer this to enriching the events with appropriate slots.

To assist in this effort, there are three ways to enrich IBM® Tivoli® Monitoring (ITM) events:

- Using Mapping Files mechanism provide by ITM
- Using SCE (State Correlation Engine)
- Using Netcool Impact product

In this paper, we will discuss how to enrich ITM Events using the ITM Product provided mapping files.

Here are the assumptions:

- IBM Tivoli Monitoring is installed.
- HUB TEMS is configured to send events to TEC (by enabling TEC Integration Facility for port 5529)
- IBM Tivoli Business Service Manager (TBSM) is installed
- Omnibus installed and configured
- EIF probe is installed and listening on a port (for e.g. 5529)

Here's an overview on how the events flow all the way from Agents to HUB TEMS, via EIF to Omnibus and eventually utilized by TBSM.

- Situations are created in ITM and distributed to Agents
- When the thresholds are violated in situations, Event is fired
- Event goes through the mapping file, as its mapping is defined, and custom slots are appended to the events
- Events reaches EIF probe and are gone through the rules file and then inserted into omnibus objectserver
- Next Event is available for TBSM to create services, indications etc.

In our example below we have 2 separate systems:

- ITM TEPS/TEMS/Windows OS Agent all are installed on same system (Windows 2003)
- TBSM, Omnibus and EIF are all installed on same systems (Windows 2003)

Starting with the mapping file on HUB TEMS, the default maping files are located under C:\IBM\ITM\CMS\TECLIB directory on the HUB TEMS.

Note: As a best practice, when adding slots for the events from ITM, create your own mapping file. It will be easy to keep track of changes as well as when the new fix packs are applied on the system, the file will not get overwritten.

The mapping file has to be 3 characters in length (before extension), just like the default ones knt.map, kux.map etc.

<id> tag is recommended to be 99 for the user defined event mapping (to identify that the Events are mapped to a custom mapping file)
mapAllAttributes="Y", will let you carry over all the default slots plus the new slots defined in you custom mapping file

situation name, specify the ones, to which you want and can use regular expression
class name, use the Attribute Groups name from which the situation is created

Here's a custom mapping file, we will use to append slots to all situations starting with BSM

```
<?xml version="1.0" encoding="UTF-8"?>
<itmEventMapping:agent
xmlns:itmEventMapping="http:
xmlns:xsi="http:
xsi:schemaLocation="http:>
<id>99</id>
<version>6.2.0</version>
<event_mapping>
<situation name="BSM*" mapAllAttributes="Y">
<class name="ITM_NT_Processor"/>
<slot slotName="Lob">
<literalString value="Finite Credit"/>
</slot>
<slot slotName="ServiceName">
<literalString value="Finite Co."/>
</slot>
<slot slotName="ApplicationName">
<literalString value="Process Credit Applications"/>
</slot>
<slot slotName="LocationName">
<literalString value="Boston"/>
</slot>
<slot slotName="OSType">
<literalString value="Windows 2003 EE"/>
</slot>
<slot slotName="HardwareVendor">
<literalString value="IBM"/>
</slot>
<slot slotName="HardwareModel">
<literalString value="XSeries"/>
</slot>
<slot slotName="SupportGroup">
<literalString value="Finite IT"/>
</slot>
</situation>
</event_mapping>
</itmEventMapping:agent>
```

You can see the below the contents of the event fired.
Highlighted in BLUE, are the slots added by the mapping file.

```
ITM_NT_Processor
cms_hostname='chafe.tivlab.austin.ibm.com'
cms_port='3661'
integration_type='N'
master_reset_flag=''
appl_label=''
```

```
situation_name='BSM_NT_Percent_Processor_Time'
situation_type='S'
situation_origin='Primary:CHAFE:NT'
situation_time='04/08/2008 16:18:51.003'
situation_status='Y'
severity='CRITICAL'
date='04/08/2008'
situation_displayitem='3'
source='ITM'
sub_source='Primary:CHAFE:NT'
hostname='CHAFE'
origin='9.48.139.136'
sub_origin='3'
adapter_host='CHAFE'
Lob='Finite Credit'
ServiceName='Finite Co.'
ApplicationName='Process Credit Applications'
LocationName='Boston'
OSType='Windows 2003 EE'
HardwareVendor='IBM'
HardwareModel='XSeries'
SupportGroup='Finite IT'
pct__dpc_time='0'
pct__interrupt_time='6'
pct__privileged_time='27'
pct__processor_time='45'
pct__user_time='19'
apc_bypasses_per_sec='0'
dpc_bypasses_per_sec='0'
dpc_queued_per_sec='56'
dpc_rate='1'
interrupts_per_sec='87'
processor='3'
server_name='Primary:CHAFE:NT'
timestamp='1080408161850875'
msg='[(%_Processor_Time>=30 AND Processor<>"_Total" ) ON Primary:CHAFE:NT ON 3
(%_Processor_Time=45 Processor=3 )]'
situation_eventdata='~'
END
```

For testing purposes to see the contents of Events (before sent to EIF/Omnibus) users can change the om_tec.config (specifies the location of EIF Probe running/listening for Events from HUB TEMS) file present under the C:\IBM\ITM\CMS\teclib directory, to dump into a file, to validate the Event.

Specify an invalid hostname for ServerLocation variable.
Restart HUB TEMS
Events when fired, will be written to the cache file (om_tec.cache) specified by the BufEvtPath variable.

```
# Component: OMEGAMON - Tivoli Event Integration
# Description: OMEGAMON - TEC Event Integration configuration file
# (C) COPYRIGHT IBM Corporation 2005
# Licensed Material - Property of IBM Corporation

# this variable points to the T/EC host where we forward events
ServerLocation=testevents
#
ServerPort=5529
#
#EventMaxSize=4096
#
RetryInterval=5
getport_total_timeout_usec=50500
NO_UTF8_CONVERSION=YES
ConnectionMode=co
BufferEvents=YES
BufEvtMaxSize=4096
BufEvtPath=./TECLIB/om_tec.cache
FilterMode=OUT
Filter:Class=ITM_Generic;master_reset_flag='';
```

Note: If there's a need to modify the mapping file, user can do that and issue this command when changes are made, without restarting the HUB TEMS.

C:\IBM\ITM\BIN>tacmd refreshTECinfo -t all

Once we determine the contents and correct slots for our Events, the next step is to add these new slots in the rules file running on the EIF Probe, so the slots in the events coming from HUB TEMS can be picked up and processed by the EIF probe rules file.
There is a rules file names tivoli_eif.rules under the C:\IBM\Netcool\omnibus\probes\win32 directory on the system where EIF probe is installed.

The tivoli_eif.rules contains the basic event processing rules used by the tme10tecad probes in support of TBSM. The file is primarily used to map event tokens to columns in the alerts.status table which is contained in the Omnibus ObjectServer database schema.

Edit the file, and you will find this section, mid way in the file.
Add the slot names in that section as shown below in BLUE in this format
@SlotName=$SlotName

```
\### Handle ITM Events

if ( exists( $situation_name ) )
{
@Identifier = $situation_name + ":" + $situation_origin + ":" +
$situation_displayitem + ":" + $ClassName
@AlertKey = $situation_name
@Node = $situation_origin
@NodeAlias = $situation_origin
@Agent = $source
@Type = 20

@ITMDisplayItem = $situation_displayitem
@ITMStatus = $situation_status
```

```
@ITMTime = $situation_time
@ITMEventData = $situation_eventdata
@ITMHostname = $cms_hostname
@ITMIntType = $integration_type

@Lob = $Lob
@ServiceName = $ServiceName
@ApplicationName = $ApplicationName
@LocationName = $LocationName
@OSType = $OSType
@HardwareVendor = $HardwareVendor
@HardwareModel = $HardwareModel
@SupportGroup = $SupportGroup

\### Populate BSM specific columns.
```

Now we will have to add the above defined slots into the Omnibus ObjectServer database.
Here are the steps to do that.

```
C:\IBM\Netcool\omnibus\bin>isql.bat \-S NCOMS \-U root
Password:
1> alter table alerts.status add column Lob varchar(128);
2> go
(0 rows affected)
1> alter table alerts.status add column ServiceName varchar(128);
2> go
(0 rows affected)
1> alter table alerts.status add column ApplicationName varchar(128);
2> go
(0 rows affected)
1> alter table alerts.status add column LocationName varchar(128);
2> go
(0 rows affected)
1> alter table alerts.status add column OSType varchar(128);
2> go
(0 rows affected)
1> alter table alerts.status add column HardwareVendor varchar(128);
2> go
(0 rows affected)
1> alter table alerts.status add column HardwareModel varchar(128);
2> go
(0 rows affected)
1> alter table alerts.status add column SupportGroup varchar(128);
2> go
(0 rows affected)
1> quit
```

Note: The Slot names defined in the mapping files, rules file and in the objectserver needs to be all same case (either lowercase, UPPERCASE or CamelCase).

Now restart the EIF probe and we will see the events in omnibus object server via Event List gui.

The next step is to create services based on these Events data in TBSM.

First we will create templates. Based on these templates, Service is created either manually or automatically by using the Event Slots.

In this example, we created templates and used auto pop rule to create Services.

The template structure is created.

The resulting Service Tree from the above defined templates. This tree is created because we used the auto pop rule, using the slots from the Event listed above

Note: the tree structure contains the name, which we added to slots in the mapping files in ITM

The ITMHostname variable will result into "chafe.tivlab.austin.ibm.com"

The ApplicationName variable will result into "Process Credit Applications"

The Lob variable will result into "Finite Credit"

For more details on how to create templates and services in Tivoli Business Service Manager, please take a look at the Scenario Guide for TBSM (IBM Tivoli Business Service Manager, Scenarios Guide, Version 4 Release 1.1, SC23-6043-01) on the information center (http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/welcome.htm)

For more details on how to use the mapping files and description of xml tags, please take a look at the "Coding an event mapping file for ITM TEC Event Forwarder" posted on OPAL website (http://www.ibm.com/software/tivoli/opal)

Event Driven Service Instantiation using Tivoli Business Service Manager and Netcool,Impact

The purpose of this paper is to show how IBM® Tivoli® Business Service Manager (TBSM) services can be restructured and used based on events from various sources. These events might be generated by other Tivoli products, such as IBM Tivoli Monitoring, or generated as the result of interrogating a data source, such as a database, on an interval basis. This paper focuses on two specific scenarios:

1. The first scenario shows how existing services in TBSM can be restructured based on transactional events to create additional service views in the TBSM Service Tree. This scenario uses a Tivoli Netcool/Impact policy to interrogate a database table on an iterative basis and generate events to populate the TBSM Service Tree with new services based on data in that database. These new services will be constructed based on other existing services.

2. The second scenario shows how events generated from IBM Tivoli Monitoring can be used to populate service instances in the TBSM Service Tree. This scenario shows how Tivoli Netcool/Impact policies and TBSM auto-population rules can be configured to create service instances based on events that are generated by Tivoli Monitoring distributed situations.

An assumption is made that the audience of this paper has basic knowledge of TBSM, Tivoli Netcool/Impact, and Tivoli Monitoring. For more detailed information about these products, refer to the References section of this paper.

This paper is not an implementation guide. It is assumed that all products are installed, including TBMS (with OMNIbus), Tivoli Netcool/Impact and Tivoli Monitoring. It also assumes that any integration between these products has been configured. Specifically, it assumes that the EIF Probe has been configured to forward events from Tivoli Monitoring to the OMNIbus ObjectServer component of TBSM.

Finally, the scenarios implemented in this paper are primarily for educational purposes. Although care was taken to efficiently implement these scenarios, production environments can vary considerably. Materials presented in this paper should be used for reference only.

To read this paper by Rob Davis, click here.

IBM Tivoli Business Service Manager V4.2 Custom Launch Integrations

IBM® Tivoli® Business Service Manager V4.2 (TBSM) includes preconfigured launch options that provide navigation to other Tivoli applications. You can also configure additional launch options to most Web applications such as other Tivoli or third-party applications.

This white paper describes the steps needed to create custom launches using a URL of your choice and to enable context from the business service you are launching from to be passed to the application you are launching as part of the URL. The two following scenarios are used to illustrate the customization required:

- A launch to Google and Google Finance from services that were manually created
- A launch to IBM Tivoli Composite Application Manager (ITCAM) SOA from services created from the Discovery Library Adapter (DLA)

Terminology

See Table 1 for definitions of key terms and concepts that are discussed in this paper.

Table 1 Terms and Definitions used in this paper

Concept	Definition
CDM	The Common Data Model requires that resource instances be named according to the naming rules of their model namespace. Every manageable resource has both an identity and a name
DLA	The Discovery Library Adapter extracts data about discovered resources and their relationship from a specific application and shares that data using Common Data Model terminology.
IDML	The Identity Markup Language is an extension of the Common Data Model XML schema developed by IBM. The IDML is used by Tivoli Discovery Library Adapters to identify configuration items, their attributes, and relationships.
ITCAM for SOA	IBM Tivoli Composite Application Manager for SOA offers integrated management tools for Web and enterprise infrastructures to aid SOA life-cycle availability and performance.
SOA	Service-oriented architecture is a method for systems development and integration where functionality is grouped around business processes and packaged as interoperable services.
TADDM and CCMDB	IBM Tivoli Application Dependency Discovery Manager stores and provides complete and detailed application maps of business applications and supporting infrastructure, including cross-tier dependencies, run-time configuration values, and complete change history. The new version of the TADDM software is called IBM Tivoli Change and Configuration Management Database (CCMDB). IBM Tivoli Change and Configuration Management Database is an integrated productivity tool and database that helps you manage, audit, and coordinate the change and configuration management processes through user interfaces and workflows that are designed to facilitate cross-silo cooperation.

TBSM	IBM Tivoli Business Services Manager delivers the technology for IT and business users to visualize and assure the health and performance of critical business services.
SCR	Service Component Repository Is the central location in TBSM that interfaces with CDM implementation (such as TADDM and DLAs) to make discovered CDM resources available for business service monitoring with TBSM.

Customization Process Overview

The customization process creates the following custom launches:

- Launch to an external web application such as Google without any specific service instance information.
- Launch to an external web application that can accept "context" via its URL parameters. For our example we will launch to Google Finance and pass equity data that is contained in the TBSM service attribute fields and is different for each TBSM service instance.
- Launch to another Tivoli product using service instance information obtained via a Discovery Library book produced by an discover library adapter (DLA)for that product. The Tivoli product must be able to be launched via a URL and be able to accept context data that was originally supplied in the discovery Library book. In our example we will launch to TEP using data supplied in Discovery Library book produced by the ITCAM for SOA Discovery Library Adapter.

To create these launches, the following processes are needed:

- Create or select the templates and services that the customized launch will apply to.
- Create a customized view definition which allows you to map personalized actions
- Edit customized view definition to add custom launch URL and specifications. These launch actions will be added to the right click menus displayed in the Service Editor or Service Viewer portlet. Note that changes to the view definition will not affect the right click menu that is displayed in the Service Tree Navigation portlet.
- If the custom launch URL uses attributes from the service model to pass instance information to launched application, you may need to enable the usage of those attributes in the view definition.
- Advanced techniques that involve manual editing of certain XML files will enable custom launch actions to be added to the right click menus displayed in the Service Tree.

The instructions in this document assume that you are already logged into the Tivoli Integrated Portal (TIP) console with a user ID that has a role of tbsmAdminUser so that you can access the Service Administration page.

Create and Select Template and Services

This white paper provides a description of services associated with stock-trading exchanges to illustrate the techniques required to create a custom launch. A description of services associated with resources created by the Discovery Library Toolkit as a result of reading an IDML (International Development Markup Language) book produced by a Tivoli product development team. Detailed instructions on how to create all the services used as examples in the paper are available in the Appendix section.

You can also choose to select existing templates and services in your installation to follow the examples in this paper. The launch to Google scenario applies to all services and you don't have to select any specific template. The launch to Google Finance scenario can be used any template you desire and you can substitute other attributes from your template in place of the ones we have set up. The launch will work but it is unlikely that the launch context passed will result in a real equity quote.

If you are wish to use existing services derived from reading a Discovery Library book (or importing data from

TADDM) that is not from ITCAM for SOA be sure to read the section on "Enabling launch related attributes derived from Discovery Library books (SOA Example)" and determine whether you have equivalent attributes in your services to implement a launch.

Create a Custom View Definition

TBSM 4.2 ships various view definitions that allow you to control the visual content that the Service Editor or Service Viewer portlet displays. The view definition also defines the actions appear as options on the Right Click menu. You can review the settings for any of the view definitions that we deliver but you cannot use the console to edit them. However, you can easily make a custom view definition based on any other existing definition which will allow edits so that you can change settings as you desire. This section will guide you in creating a "MyRelationships" which is a custom view definition based off the Relationships view definition which is the default view that the Service Viewer opens with. If you need any additional help, please refer to the TBSM Service Configuration Guide, Chapter 26 "Custom view definitions"

1. From the Service Navigation pull down menu, select **Services** and then click on the Online Services node in the Service Tree.
2. You should see the current view definition listed in pull down selection field (Item 2*)**. If you don't see this, click on the View menu item (Item 1), scroll over the Toolbars option and enable the View Definition choice. *Relationships* is the default view definition, but if it is not selected, use the drop down menu to open the list of available view definitions and select*Relationships*. The Service Viewer will reload changing the view shown. Once that is completed, click on *Edit View Definition** button (Item 3).
3. The Edit View Definition window opens showing the setting for the Relationships view. Click on the **Save as New** button on the bottom of the screen. A popup window will appear where you can type in*MyRelationships* as the name of the new custom view definition you want to create. It will have the same settings as the *Relationships* view.
4. When you click **OK** on the popup window, the Service Editor will refresh the visual display in Viewer and use *MyRelationships* as the view definition. It looks identical to what was shown for *Relationships* since we have not yet changed any settings.
5. View definition settings are kept as xml files in the $TBSM_DATA_SERVER_HOME/av/xmlconfig directory. The file name follows a pattern of
ViewDefinition_*xxxx*.xml where *xxxx* is the name you assigned to the view definition.
When you created MyRelationships view definition, the file ViewDefinition_MyRelationships.xml was also created

Enable service attribute data for configuring launch actions

In addition to action configuration settings, view definition files also configure which service attributes are available for actions. You can find which service attributes are available for a specific service by viewing the service in the Service Editor Portlet and opening the **Edit Service xxxx** tab. Scroll down and select the **Additional** tab to view the attributes. These attributes might contain data that can be used to form part of the launch URL (such as parameter data) or to set conditions to determine whether an action is applicable for a specific service. If an action is not applicable, then it can be disabled and grayed out so the user cannot select it. By default, all the attributes defined in templates are *not* available for actions. You must enable those attributes that you want to include in your custom launch actions. Because enabled attribute data is kept in memory for the client model of the Service View for every service instance, enable only those attributes you will truly use and not every attribute defined in templates.

This section guides you in enabling service attributes to be used for launch actions. If you need any additional help, see the "Using additional properties in right-click actions" section in the "Custom Settings" chapter of the *IBM Tivoli Business Service Manager Customization Guide*.

FieldToPassToModelExpr XML element

The *fieldToPassToModelExpr* xml element is used to identify an attribute that can be used by the view definition. The element form is:

<fieldToPassToModelExpr modelField = "*AttributeName*">*AttributeValue*
</fieldToPassToModelExpr>

where *AttributeName* is the name of the attribute you want to pass and *AttributeValue* is the value for the attribute. You can use this format to pass an attribute for all services with a set value such as this:

<fieldToPassToModelExpr modelField = "myhost">myanez </fieldToPassToModelExpr

Generally, however, you want the attribute to contain a value from the service you are performing the action on. In that case the *AttributeName* must match an attribute defined in the service template and the*AttributeValue* must contain the *AttributeName* string. In other words, the *AttributeName* and *AttributeValue* fields must match. Wherever *AttributeName* is used in an action and if the service does not have an*AttributeName* attribute, then a value of null is returned.
Changing or adding *fieldToPassToModelExpr* information in any of the view definition files requires a recycle of the Business Service Manager data server for the change to take effect, so it best to do all the updates you need at the same time and then recycle the server.

Enabling service attributes for Trading Exchange Service

Now you want to use the following procedure to enable the *MostActiveEquity* and *TopGainer* attributes that are required for your GoogleFinance launches:

1. Shut down the TBSM data server.
2. Open the ViewDefinition_MyRelationships.xml with the editor of your choice and search for "fieldToPassToModelExpr." You will find a block of fieldToPassToModelExpr xml statements within an outer dataTypeMapping xml element similar to this:
   <dataTypeMapping dataTypeName="ServiceInstanceBean">
   ......
   <fieldToPassToModelExpr
   modelField="IBM_Tivoli_Monitoring_Services_sourceContactInfo">
   IBM_Tivoli_Monitoring_Services_sourceContactInfo
   </fieldToPassToModelExpr>
   </dataTypeMapping>
3. Add the following statements after the last __fieldToPassToModelExpr and before the </dataTypeMapping> tag:
   <fieldToPassToModelExpr modelField="MostActiveEquity">MostActiveEquity
   </fieldToPassToModelExpr>
   <fieldToPassToModelExpr modelField="TopGainer">TopGainer
   </fieldToPassToModelExpr>

   The AttributeName must match exactly the display name of the attribute in the template (which is case-sensitive). Save and close the view definition file.
4. Start the TBSM data server. (If you plan to enable more attributes for the SOA example, do not restart the server yet).

Enabling service attributes for SOA services

As mentioned previously, a DLA book provides you with data on resources and their relationships that can be used to build TBSM services. There are key items that TBSM imports from a DLA book and keeps as service attributes that are critical to enabling a launch from a DLA-derived service back to the source product. These items are:

- The *sourceContactInfo* attribute is supplied as part of the ManagementSoftwareSystem class. The product generating the DLA identifies itself through the ManagementSoftwareSystem tags. In many cases, the sourceContactInfo tag identifies the URI (host name and port) of the server that supports a Web console, which can be used by TBSM as the target host system to launch back to. However, the sourceContactInfo URI varies across products. Some products supply a host name and port of a server that cannot be launched to. In other cases, the sourceContactInfo URI contains more than the host name and port and can be set up as a launchable URL. Some products allow the sourceContactInfo data to be configurable by the user.

  You need to understand exactly what is being provided by DLA for this field and determine if you can use that to launch.
  Note that if you can configure this field, the ideal value for a TBSM launch use is simply the URI with the fully qualified host name and port number of the UI server. A fully qualified host name (with port) rather than a short host name or IP address is required if you plan to use Single Sign On (SSO) support to eliminate multiple login requests when navigating across applications. SSO is configured based on domain names, which is why a fully qualified host name (containing domain) is required.

  An example of this concept of configuring a TBSM launch point is the following IBM Tivoli Monitoring sourceContactInfo value:

  <cdm: Process.ManagementSoftwareSystem sourceContactInfo=http://meywin1.raleigh.ibm.com:1920>

- The *sourceToken* attribute for a resource is the ID that this management software uses to identify a particular instance in terms that it can understand. Here is an example from a Tivoli Monitoring DLA:

  <cdm:sys.windows.WindowsOperatingSystem id="meywin1.raleigh.ibm.com-WindowsOperatingSystem" sourceToken="managed_system_name=Primary:MEYWIN1:NT& object_id=p@Primary:MEYWIN1:NT">

  Often the sourceToken can be used in the launch URL to pass the context for the launch.

  Because a TBSM service can be a composite service that reconciles data from multiple DLA books, it can retain many sourceContactInfos and sourceTokens attributes. In other words, the same resource is being monitored or managed by multiple products, each producing their own IDML book. Therefore, attribute names for these items are prefixed with a product name (such as IBM_Tivoli_Monitoring_Services_sourceContactInfo and IBM_Tivoli_Monitoring_Services_sourceToken). Note that blanks in the product name are replaced with underscores.

  If a TBSM service has sourceContactInfo and sourceToken data, that data can be seen in the **Edit** tab of the Service Editor portlet. Click the **Additional** tab and scroll through the attributes to find this data.

  In this example, the AcmeService child services is imported from the IBM Tivoli Composite Application Manager (ITCAM) for SOA DLA book. The ITCAM for SOA DLA V7.1 uses the **sourceContactInfo** field for the Tivoli Enterprise Monitoring Server (TEMS or monitoring server) and not the Tivoli __Enterprise Portal Server (TEPS or portal server). You cannot use this data to launch a Tivoli __Enterprise Portal (TEP or portal) browser client if your monitoring server and portal server are on different computers. In this example, both servers are on the same computer so the sourceContactInfo can be used. Note that future versions of the SOA DLA will allow a user to configure the sourceContactInfo to the TEPS.

  Although the ITCAM for SOA DLA defines several cdm resources, only the cdm:soa.WSEndpoint objects have associated sourceTokens and only the TBSM services that are mapped from those definitions include ITCAM for SOA sourceToken attributes. TBSM creates business service instances from the other definitions but these services do not have an ITCAM sourceToken and therefore cannot be used to launch back to the ITCAM for SOA

workspaces in the Tivoli Monitoring application.
You can now enable the **IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceContactInfo** __and the **IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceToken** attributes for use in launch definitions.

1. If started, shut down the TBSM data server.

2. Open the ViewDefinition_MyRelationships.xml file with the editor of your choice and search for "fieldToPassToModelExpr." Find the block of fieldToPassToModelExpr xml statements within an outer dataTypeMapping xml element similar to this:
```
<dataTypeMapping dataTypeName="ServiceInstanceBean">
......
<fieldToPassToModelExpr
modelField="IBM_Tivoli_Monitoring_Services_sourceContactInfo">
IBM_Tivoli_Monitoring_Services_sourceContactInfo
</fieldToPassToModelExpr>
</dataTypeMapping>
```

3. Add the following statements after the last **ToPassToModelExpr** __field and before the </dataTypeMapping> tag:
```
<fieldToPassToModelExpr modelField =
"IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceContactInfo">
IBM_Tivoli_Composite_Application_Manager_for_SOA__sourceContactInfo
</fieldToPassToModelExpr>
<fieldToPassToModelExpr modelField=
"IBM_Tivoli_Composite_Application_Manager_for_SOA__sourceToken">
IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceToken
</fieldToPassToModelExpr>
```

   The AttributeName must match exactly the display name of the attribute in the template (the name is case-sensitive). Save and close the view definition file.

4. Start the TBSM data server. (If you plan to enable more attributes for the SOA example, do not restart the server yet).

   Create Launch Actions

   Using the custom view definition you created in section 4, you can now edit the associated actions and add additional items. The actions you change can be associated with a specific template so that only those services that use that template will see the actions you add or change. You can also choose to have the actions you change reflected in every template.

   To start our edit session for actions, reopen the Service Viewer portlet (open the Service Administration page, select **Services** for the navigation view, and click on any service). Select the MyRelationships custom view definition. After it has finished reloading, click the **Edit View Definition** button again that shows the editable settings for the MyRelationships view. Now click on the **Actions** tab.

   You will be creating a couple of different launch actions, but all actions use the following steps:

1. Select **Add New Child Wizard** __in the **Edit Action** field to define a new launch action.

2. Indicate either a specific service template to work with or that you will operate on defaults set for all templates. The pop-up window is updated to show the current actions for your selection. In particular, the Right Click Menu Option table will be filled in.

3. Click the **New** button that adds a new action at the bottom of the Right Click Menu Option table. Change the

default value of the action to **Add New Child Wizard** for your new launch action.

Creating a static launch action available from all services

The static launch action is a very simple launch use case where you launch to another application using a URL. You may not need to pass specific service-instance data as parameters or context to the application you are launching. In this example, you launch to Google Website (www.google.com):

1. Define the new launch action with parameters indicated in Figure 4. Be sure to type in the full URL needed to reach the application. Use the **Action Frame** field to choose whether to open a new browser window (or tab) each time the launch action is clicked or to re-use an existing window that had been already opened for this launch action. If you use " _blank", a new browser window is opened each time the**Launch Google** action is selected. If you supply a name such as GoogleWindow each time Google is launched, a search for a browser window with that name is made. If none is found, a new browser window is opened and assigned that as its name. Otherwise, if a browser window with that name is found, it is re-used to launch the action.
2. Click the **Set Defaults** button to designate it as launchable from every service because Google is very useful.
3. After adding a new action to the Right Click Menu Options table, change the value from **Add New Child Wizard** to **Launch Google.**
4. Click **OK** to save your changes and return to the Service Viewer.
5. Right-click on any service shown in the viewer to see your new action at the bottom of the menu. When you click it, Google is launched.

Creating a dynamic launch action available from selective services

Most launch scenarios are not as simple as our Google launch example. Generally, you might want to pass some data contained in the service you are viewing to the application you are launching. You can make a URL dynamic by using a substitution variable within the URL string that references attribute data from that service.
In Section 5, you noted that the view definition files contained **fieldToPassToModelExpr** XML tags that defined which service attributes can be used in action definitions. You added and enabled some new attributes (such as MostActiveEquity) so that you can use them in your custom launches.

To indicate that you want to substitute the data from a particular attribute at some point in the URL, you use a substitution variable reference. This reference is the attribute name with a double underscore appended before and after the name, such as __*attributeName*__. If the attribute is not contained in the selected service or has not been enabled in the view definition, 'Null' is substituted in the URL string.

In this example, you launch to GoogleFinance from any of the Trading Exchange services to see detailed data for the equities that are most active on that exchange. The Google Finance Web site (finance.google.com/finance) can accept stock symbols as input parameter and your custom attributes are expected to contain stock symbols so they can be used in the URL.

1. Define new launch action with parameters indicated in the following table.

Action Name	**launchGoogleFinance**
Action Display Name	Show details on Most Active Equity
Action Description	launch Google Finance for most active equity
Action URL	http://finance.google.com/finance?q=__MostActiveEquity__
Action Frame	GoogleFinanceWindow

2. Because this launch uses an attribute that has been defined only in the TradingExchanges template, limit where that launch option is displayed. Instead of choosing the **Set Defaults** button that you chose for the basic Google launch, indicate the specific template that you want this action to be associated with.
3. After adding a new action to the Right Click Menu Options table, change the value from **Add New Child Wizard** __to **Show details on Most Active Equity**.
4. Click OK to save your changes and return to the Service Viewer.
5. If you now right-click either of the two trading exchange services shown in the viewer (NY Stock Exchange or Nasdaq), the new action is displayed at the bottom of the menu. If you right-click on any other service such as Online Services that is not based on the TradingExchanges template, that action is *not* visible. You can also see different launch results depending on which trading exchange service you launch from. The NY Stock Exchange launches Google finance, which shows detailed quotations for IBM because that is the value listed in the **MostActiveEquity** field. The launch from Nasdaq shows details on Intel.

   The previous example uses only one attribute in the action URL but you are not limited in how many attribute substitutions you can make. At the Google Finance site, you can supply multiple stock symbols that results in a comparison graph for those equities. You can also create another launch entry to compare the most active equity to the equity that had the top gains. Using this same technique, you can achieve similar results for different action display names and action URLs.
   The URL contains the following two attributes: http://finance.google.com/finance?q=__MostActiveEquity__\+__TopGainer__
   If you launch the URL from the Nasdaq service, you see a Google finance comparison chart showing both Intel and Baidu data, which represent the values set in that service for those attributes.

   TIP: If you have a launch action with attributes and you see that the URL launch includes a 'NULL' value where you expect an attribute value, check for the following information:

- Does that specific service contain that attribute and does it have a non-null value?
- Does the view definition file for the view you are using have a **fieldToPassToModelExpr** tag that enables that attribute? Is the attribute name identical to your action URL and identical to the attribute name in the template?
- Have you restarted the TBSM data server after making the changes to the view definition file?

Creating a launch to another Tivoli product associated with a service

You have seen that you can create TBSM services from the information supplied by another Tivoli product using a discovery library book. You can potentially use that information to create launches back to that product's GUI if views pertaining to those services are available. If you want to launch to another product, the first thing you must determine is the URLs they support (status or task entry points) and specifically the parameter data they require. Then determine if that parameter data has been supplied in the discovery library book and is accessible as a service instance attribute. Generally (but not always) the sourceToken attribute supplied for a resource is part or all of the parameter data needed in the URL.

Your goal is to form a launch URL by concatenating the sourceContactInfo + the status entry point + sourceToken attributes.

In this example, you want to launch to the ITCAM for SOA related workspaces in the portal console. The ITCAM product team has indicated that the Application Server Services Management or the Services Management Agent Environment workspaces can be accessed using the http:// TEPServerhost: Port///cnp/kdh/lib /cnp.html?managed_system_name= managedSystemName __ URL. Other workspace views that are set up by ITCAM for SOA might not be accessible using a URL. Often, single views from a URL that a product provides are not accessible.

Looking at the data supplied in the discovery library book, you see that a WSEndpoint definition that is associated with an application server has a sourceToken that contains the managedSystemName. In fact the sourcetoken format is "managed_system_name= *managedSystemName."* You can use the sourceToken in your URL formation.
If this is a ITCAM for SOA V7.1 DLA, you must verify whether the sourceContactInfo data identifies the portal server and port (in a split TEPS/TEMS system, this is not true). If it is, you have enough data (the URL, the server host: Port, and the managed system name using the sourceToken) to launch back to the ITCAM for SOA workspace in the portal console. Although you derived other service instances (such as the SOA operations and computer systems) from the DLA book, there is no usable attribute data or accessible URL panels to launch to using these services. If you expand a section of the SOA hierarchy, only the application servers (circled in red) can be used for launch purposes. If you use the **Edit Service** tab, you can see the **IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceContactInfo** and**IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceToken** fields.

Use the following procedure to create the launch:

1. Define a new launch action with the parameters indicated in the table. Note that you are using substitution variables (indicated in blue) in the URL to bring in the ITCAM for SOA sourceContactInfo and source token data. In the action frame, you must use CNPWindow_1 if you want to re-use an existing portal browser window. You cannot use a string of your own devising because the portal overrides your setting and forces the window name to the internal name (such as CNPWindow_x, where *x* represents each portal window or tab that the user has overtly opened in the portal console).

Action Name	**SoaLaunch**
Action Display Name	Show Application Server (ITCAM for SOA)
Action Description	launch TEP application server workspace
Action URL	__IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceContactInfo__///cnp/kdh/lib /cnp.html?__IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceToken__
Action Frame	CNPWindow_1

2. Although this launch uses attributes that are not in every template, attributes are not strictly defined in a single template. For now use the Set Defaults attribute to allow this launch option to be displayed for all service instances. Use a different technique later on to enable this option only for those services that actually have the attributes you need.

3. After adding a new action to the Right Click Menu Options, change the value from **Add New Child Wizard** __to **Show Application Server (ITCAM for SOA)**.

4. Click **OK** to save your changes and return to the Service Viewer.

5. Right-click on any service shown in the viewer to see your new action at the bottom of the menu. If you right-clicked it from the any of the application servers instances (such as server1 under the getEnterpriseCustomer{http://ec.retail.samples.wsm.ibm.com\} node) and selected the option, it launches to the portal successfully and opens the Application Server Services Management workspace. However if you right-clicked a different type of service, such as the getEnterpriseCustomer{http://ec.retail.samples.wsm.ibm.com\} icon

that is mapped to a Websphere operation, and selected the launch, it fails and you see that the attempted URL looks similar to "NULL///cnp/kdh/lib/cnp.html?NULL." The service did not contain the attributes you needed so that launch was not appropriate for that service and you should not be allowed to select it.

Conditionally enabling and disabling a launch action

To resolve the problem you encountered with the SOA launch, create an action that is available only when the attribute data is there. By directly editing the action definition file, you can accomplish this task.

When you create a new action, an XML file describing the action is created and inserted into an existing file called canvasOpenURLActions.xml located under TBSM_DATA_SERVER_Home\ av\xmlconfig_._ You can edit the XML directly to add tags that enable the actions only when a conditional expression is met.
IMPORTANT: Be sure to back up this file before making the manual changes in this procedure for creating a launch action:

1. First locate the SOA launch action created by searching the file for "SoaLaunch" (or whatever value you assigned to the actionName) to find the following XML definition:
```
<openURLAction description="Show Managed System for SOA Server"
displayName="Show Application Server (ITCAM for SOA)"
enableDisableExpression=""
name="SoaLaunch"
permissionCheckerClassName="com.micromuse.sla.map.AVCheckRADInstancePermissionsImpl"
roleRequired="tbsmViewService" target="CNPWindow_1" visibleInGUI="true"
>__IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceContactInfo__///cnp/kdh/lib
/cnp.html?__IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceToken__
</openURLAction>
```
2. The enableDisableExpression tag can be used to add a conditional expression that determines whether the action is enabled (the user can select it from the right-click menu) or disabled (visible but grayed out and non-selectable). You do not have to check for both attributes (sourceContactInfo and sourceToken) because the toolkit always creates both if a sourceToken is available from a discovery library book. In this example, change the value of the enableDisableExpression attribute to determine if one of the attributes exists in the following specific service instances and then save the file:
```
enableDisableExpression = "'__IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceContactInfo__' !=
'NULL'"
```
3. Any time you manually edit the canvasOpenURLActions.xml file (or any of the files in that directory), you must stop and restart the TBSM data server for the changes to take effect. When that is done, go back into the service viewer for main AcmeService or any service that is not one of the SOA application servers and view using the MyRelationships view definition. Right-click he service icon to see the "Show Application Server (ITCAM for SOA)" option at the bottom of menu, which is grayed out.
4. Right-click one of the application server icons (server1) to see the option enabled.

Alternate Launch Actions

As indicated previously in this paper, the V7.1 ITCAM for SOA DLA supplies the monitoring server host for the sourceContactInfo attribute. If you have set up the monitoring server and portal server to be on different systems, the launch action used previously does not work because there is nothing in the discovery book that supplies the portal server information. This situation may also be true for other Tivoli products that might include a back-end server instead of a user-interface server as the sourceContactInfo attribute.

The following workarounds might help you launch to the V7.1 ITCAM for SOA product:

- If there is a single UI host server that recognizes all services related to that product, you can explicitly specify the host name and port as part of the URL action instead of using a substitution variable. For the ITCAM for SOA example, if the portal servers are all linked to a single monitoring server (producer of the discovery library book) and the myanez system is a portal server, you can change the action URL to the following URL: http://myanez.raleigh.ibm.com:1920///cnp/kdh/lib/cnp.html?__IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceToken__
  You also need to change the enable expression to check for the ITCAM for SOA sourcetoken rather than the sourceContactInfo attribute.
  Changing the URL in this way might not work if there are multiple portal servers that are linked to different monitoring servers. Services derived from one discovery book are not recognized in the portal server that is linked to a different monitoring server.

- If you have multiple UI host servers that manage different domains, you might want to manually edit the **sourceContactInfo** field in the discovery library books before reading them in with the TBSM toolkit. You can change the field to reflect the appropriate UI host and port to use. Use this technique only if the toolkit is reading the books directly and not if the books are first being read into TADDM and then imported into the TBSM software. If you used this technique, be sure to update the books each time you install a new version because the **sourceContactInfo** field is continuously refreshed with the values from the latest version.

- Lastly, you can create your own attribute field to contain the UI host name and port info in the template for the resource type you want to launch from (such as BSM_Node, BSM_Application, and BSM_AppServer). Use that attribute as a substitution variable in the action URL. After services are created from a book, you can manually update that field to reflect the appropriate values. The other two techniques are preferable if your environment allows it, but this is one approach that allows you to launch if you are in a multiple UI host environment and importing data from TADDM software. If the number of services you need to support is small, this technique can be very viable, but as they grow larger, the manual editing can become cumbersome.

Extending a custom launch beyond a custom view definition

In all the preceding examples, you have created a custom launch within a custom view definition and essentially used the View Definition Editor to define and edit the new launch action. This technique works well, but has the following limitations:

- The new launch actions are visible only when you obtain the right-click menu from the Service Viewer portlet. If you right-click the same service in the Service Tree within the Service Navigator portlet, the customized launch actions do not show up.

- You cannot insert a new custom launch action in an existing submenu. You must insert the new action at the top level of the right-click menu. The TBSM program has a **Launch to** submenu that currently contains the launches to other products or applications. Some users prefer to see all launches in that submenu rather than as separate items that can be interspersed with other non-launch custom actions.

- You need to create a copy of any existing view definition your users use. Add the custom launch to each of those copies because your view definitions cannot be edited online. You must switch to those views when launching or change thei default view to automatically use one of the custom views.

You can avoid these limitations by directly editing the XML files that control action behavior in TBSM_DATA_SERVER_Home\ av\xmlconfig __directory. Directly editing the XML files was briefly referenced earlier in this paper when conditional enablement of launch actions was defined. Now you can learn about more changes you can manually make to alter your right-click menus in both the Service Viewer and Service Tree portlets.

IMPORTANT: These techniques are advanced. All manually changed files must be backed up first. If you introduce an error in your XML file, you might not be able to see some or all of your right-click menu options, so introduce each change independently so that you can troubleshoot any errors.

Before learning about the changes you can make, you must first understand the files affecting launch actions:

- The canvasOpenURLActions.xml file contains the xml definitions of each action. Other files refer to actions defined here using the *actionName* reference.
- The canvasDynamicSubMenuActions file defines a submenu and the actions that are included in that submenu.
- The treeTemplates.xml file defines right-click menu options available for service instances in the Service Navigation tree view.
- The ViewDefinition_xxx.xml files that are associated with a single view definition define which actions are available in right-click menu for services displayed
in the Service Viewer portlet.

All the following techniques assume that you have first successfully created a launch action in a custom view. The launch action is completely defined in the canvasOpenURLActions.xml file and you do not need to update that file any further.

Displaying a new launch action in the Launch to submenu

Displaying a new launch action is one of the simpler changes to make. You can see the new launch action in the **Launch to** submenu shown in *both* the Service Viewer and Service Navigator Portlets.

1. Edit the canvasDynamicSubMenuActions file and and search for "IntegrationTools" to find the following XML definition:
```
<dynamicSubMenuAction
name = "IntegrationTools"
displayName = "Launch to"
description = "Launch Integrated Products from Selected Instance."
......
<!-- Old Menu Items -->
<nextAction name = "ShowManagedSystem"/>
<nextAction name = "ShowHOPViewLocal"/>
<nextAction name = "ShowHOPViewRemote"/>
<nextAction name = "ShowPhysicalTopology"/>
<nextAction name = "ShowChangeHistory"/>
<nextAction name = "ShowCIDetails"/>
<nextAction name = "ShowOpenServiceRequest"/>
<!-- End: Old Menu Items -->

</dynamicSubMenuAction>
```

1. Add a new nextAction tag and use the actionName of the new launch action for the name field. Place it relative to the existing nextAction tags in the order that you want to see it on the submenu. In this example, you want to move the Launch Google and ITCAM for SOA launch to the launch submenu. You want to keep the portal-related launches together so you add the SOA launch after the current portal launch for managed systems. The Google launch is added to the end:
```
<!-- Old Menu Items -->
<nextAction name = "ShowManagedSystem"/>
<nextAction name = "SoaLaunch"/>
```

```
<nextAction name = "ShowHOPViewLocal"/>
<nextAction name = "ShowHOPViewRemote"/>
<nextAction name = "ShowPhysicalTopology"/>
<nextAction name = "ShowChangeHistory"/>
<nextAction name = "ShowCIDetails"/>
<nextAction name = "ShowOpenServiceRequest"/>
<nextAction name = "ShowManagedSystem"/>
<nextAction name = "LaunchGoogle"/>
<!-- End: Old Menu Items -->
```

2. For the LaunchGoogle action, the example does not introduce any attributes that were used as substitution variables but you do want to introduce new attributes for the SOA Launch. Any attributes that you enable in the custom view definition (ViewDefinition_MyRelationships.xml) must be enabled in all the other view definition files. You can open the custom view definition file, copy the xml statements from that file into the appropriate locations in the other files, as shown here:

```
<fieldToPassToModelExpr
modelField="IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceContactInfo">
IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceContactInfo
</fieldToPassToModelExpr>
<fieldToPassToModelExpr
modelField="IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceToken">
IBM_Tivoli_Composite_Application_Manager_for_SOA_sourceToken
</fieldToPassToModelExpr>
```

3. After you save your changes, stop and restart the TBSM data server.

4. Open the service tree and right-click a service instance and then open the **Launch to** submenu. The new options are displayed. Note that the **Show Application Server (ITCAM for SOA)** launch may be enabled or disabled depending on which service instance you clicked. Test the launches and confirm they are still operational.

5. Double-click the service to have the service displayed with the Service Viewer. Ensure that you are using one existing definitions (such as B). Right-click the service and then open the **Launch to** submenu. Again verify that the new launch options are displayed and operational if appropriate. You must loop through this verification for each of the views you changed.

   Anything you add to the Launch to submenu is dislayed for every service irregardless of template. If you have a launch that is defined to a specific template, such as the launch to Google finance, you might not want to put it into this folder. If you recall, that launch was dependent on attributes that were available only through one template, and produce a failed launched for other services not using that template. If you still want to have that type of launch in the submenu, be sure to add conditional enableDisableExpression tags to the action definition to check for the presence of the required attribute.

Displaying a new launch action in the standard views

You may want to retain a new launch action on the top level of the right-click menu or keep it restricted to a specific template, but still want to have it available on the view definitions included with the TBSM product (and not a custom view definition).

Steps for Launch actions applicable to all templates

1. Open the custom view definition file and search for the actionName of the new launch option. Look for this launch option within an XML section similar to this:

```
<templateMapping primaryTemplateName="DefaultTag">
<expandCollapseTemplate>false</expandCollapseTemplate>
<expandCollapseInitiallyTemplate>true</expandCollapseInitiallyTemplate>
<groupingField>PrimaryTagName</groupingField>
<indicatorMapping visualRepresentation="RADPrototype"/>
<actionMapping actionName="SingleClickedOnService" clickType="~singleClicked"/>
<actionMapping actionName="ShowRawEventsTableView" clickType="~doubleClicked"/>
....
<actionMapping actionName="launchGoogle" clickType="~popupMenu"/>
<actionMapping actionName="SoaLaunch" clickType="~popupMenu"/>
</templateMapping>
```

The *DefaultTag* value for the primaryTemplateName indicates these actions are the default for all templates. Copy the actionMapping statements for the new launch options

2. Edit each of the standard view definition files (such as Relationships) and find the XML statements for the DefaultTag templateMapping. Copy the actionMapping statements for the new launch options to the end of the other actionMapping statements in this block. Note that the order of the statements is reflective of the visual order of the option in the pop-up menu so you may position them in the way you prefer.

Steps to make Launch actions applicable to a specific template

1. Open the custom view definition file and search for the actionName of the new launch option. You can find it within an XML section similar to the following section where the primaryTemplateName reflects the name of the template you associated the launch with:

```
<templateMapping primaryTemplateName="TradingExchanges">
<actionMapping actionName="SingleClickedOnService" clickType="~singleClicked"/>
<actionMapping actionName="ShowRawEventsTableView" clickType="~doubleClicked"/>
<actionMapping actionName="ShowServiceInstanceEditor" clickType="~popupMenu"/>
<actionMapping actionName="InstantiateOneHopServiceMap" clickType="~popupMenu"/>
<actionMapping actionName="ChooseChildrenTool" clickType="~popupMenu"/>
<actionMapping actionName="ShowMemberTemplates" clickType="~popupMenu"/>
<actionMapping actionName="ViewTools" clickType="~popupMenu"/>
<actionMapping actionName="IntegrationTools" clickType="~popupMenu"/>
<actionMapping actionName="MaintTools" clickType="~popupMenu"/>
<actionMapping actionName="launchGoogle" clickType="~popupMenu"/>
<actionMapping actionName="launchGoogleFinance" clickType="~popupMenu"/>
<actionMapping actionName="launchGoogleFinanceCompare" clickType="~popupMenu"/>
</templateMapping>
```

2. If this is the first time you created actions tied to this specific template, you must copy the *entire* templateMapping definition block. Open each standard view definition file, locate the *DefaultTag*templateMapping definition block, and add the new block after the default. If you have already added actions to this specific template, the standard view definition file should already contain the templateMapping definition block for that template. Simply add the new actionMapping statements to the block in the order you want them shown in the right-click menu.

Common Steps

1. While you are still in each standard view definition file, add any attributes that you used as substitution variables in the action URL (see step 3 in section 6.4.1 for more details). Then save and close the file.

2. Save and close all files. Then stop and restart the TBSM data server for the changes to take effect.

3. Open the Service Viewer and use one of the standard view definitions (such as Relationships) to view a service instance. Verify that when you right-click the service instance icon, you see the new launch action on the menu. If the action is tied to a specific template, be sure to check that it is displayed only on service instances using that template.

Displaying a new launch action in the Service Navigator tree view

You may want to use a new launch action within the Service Navigator portlet when accessing a service in the service tree view. In section 6.4.1 "Displaying a new launch action in the "Launch to" submenu," you saw that including the new launch action within the **Launch To** submenu provides a launch from the service tree.
Use the following steps to also provide a launch from the service tree for new launch actions that are on the top level on the menu and are not within a submenu. However this technique does *not* limit an action to a specific template. If your action depends on attribute data that is not available on all services, be sure to add conditional enableDisableExpression tags to the action definition to check for the presence of the attributes needed.

1. Open the treeTemplate.xml file and search for the *<treeTemplate name = "ServiceInstance">* XML section. You should find in within an XML section similar to the following section that is within the treeTemplate block:
   ```
   <templateTreeMapping primaryTemplateName = "DefaultTag">
   .......
   <actionMapping
   clickType = "~popupMenu"
   actionName = "ViewTools"/>
   <actionMapping
   clickType = "~popupMenu"
   actionName = "IntegrationTools"/>
   <actionMapping
   clickType = "~popupMenu"
   actionName = "MaintTools"/>
   </templateTreeMapping>.
   ```
2. Add a new actionMapping tag for the new launch action you want displayed on the menu. Place it relative to the existing items in the order you want it displayed. In this case, you are adding the launch Google action after the Maintenance submenu.
   ```
   ...
   <actionMapping
   clickType = "~popupMenu"
   actionName = "MaintTools"/>
   <actionMapping
   clickType = "~popupMenu"
   actionName = "launchGoogle"/>
   </templateTreeMapping>
   ```
3. Save the file. Then stop and restart the TBSM data server for the changes to take effect.
4. Open the Service Navigator portlet and verify that when you right-click the service instance in the tree, you see the new launch action on the menu:

   The preceding section concludes our discussion on how to create custom launches to other applications from TBSM context menus.

Appendix

Create Template and Services

This section indicates how you can create both the Trading Exchanges Services and an ITCAM for SOA services samples used in this white paper. If you need additional help in creating similar services see the *IBM Tivoli Business Services Manager Service Configuration Guide*.

Create the Trading Exchange Services

The services you build in this section create a two-level hierarchy:

- The higher level is a service model that contains all possible online services.
- The lower level is a service model that specifically models trading exchanges and keeps track of the most active and top gainer equities for that day.

These services are very simple services sufficient for the purposes of this paper but by no means robust examples of the powerful, dynamic service models that the TBSM software can create.

Create Service Templates

Create the low-level template first by completing the following steps:

1. In the Task List at the left side of the console, select **Administration** -> **Service Administration** to open the Service Administration page. The **Service Navigation** portlet on that page shows the**Templates** portlet by default.
2. Click the **Create New Template** button. The **Edit Template** tab opens in the *service editor*.
3. Type in TradingExchanges __for the **Template Name** field and then type anything you want for **Description**. __
4. Optionally, click the **Display Icon: Browse** button to open the Browse Icons window and select the icon you want for TradingExchanges. All services assigned to this template include this icon.
5. Click the **Additional** tab. Then click the **New Parameter** button repeatedly to create the following three parameters, leaving the **Default Value** field blank each time.

- *Exchange*
- *MostActiveEquity*
- *TopGainer*

6. Click the **Save** button in the **Edit Template** tab tool bar.

Next, create the Online_Services template and its dependencies using the following steps:

1. Go back to the **Templates** portlet in the **Service Navigation** portlet and click the **Create New Template** button. The **Edit Template** tab opens in the *service editor*.
2. Type in Online_Services __for the **Template Name.**
3. Optionally click the **Display Icon: Browse** button to select the icon you want for Online_Services.
4. Click the **Rules** tab. Then click the **Create Good, Marginal, Bad Aggregation Rule** button . The rule you create specifies that Online_Services services are dependent upon the child services of the TradingExchanges template and aggregates their status for the parent service.
5. Enter the *OnlineServiceStatus* for the **Rule Name** field*,* select *TradingExchanges* from the menu for the **Child Rule/Mapping** field, and select the **% of children** radio button for **Condition Type**.
6. Click the **Save** button in the **Edit Template** tab tool bar.

Create TradingExchanges and Online_Services Services

Now you are ready to create services using the templates created in the previous section. Again you start with the lower-level TradingExchanges services.

1. From the Service Navigation menu, select **Services,** and then click the **Create New Service** button. The **Edit Service** tab opens in the *service editor*.
2. In both the **Service Name** and **Display Name** fields, type **Nasdaq**.
3. In the **Templates** tab, click the **TradingExchanges** template in the **Available Templates** list and click the **>>** button.
4. Click the **Additional** tab. Specify the field values for the parameters you specified in the TradingExchanges template. Enter these field values for the following parameters:

- Exchange - *NASDAQ*
- MostActiveEquity - *INTC*
- TopGainer - *BIDU*

5. Click the **Save** button in the **Edit Service** tab tool bar.
6. Repeat Steps 1 thru 5 to create a second TradingExchanges service but use these values:

- Service Name - *NYStockExchange*
- Display Name - *NY Stock Exchange*
- Exchange - *NYSE*
- MostActiveEquity - *IBM*
- TopGainer - *GE*

Note that for simplicity you are manually supplying values for the equity parameters; in reality, this type of attributes is most likely dynamically changing based on incoming events or rules.

Lastly, create the higher-level Online Services service using these steps:

1. From the Service Navigation menu, select **Services,** and then click the **Create New Service** button. The **Edit Service** tab opens in the *service editor*.
2. For **Service Name**, type *Online_Services*. For **Display Name,** type *Online Services*
3. In the **Templates** tab, click the template *Online_Services* in the **Available Templates** list and click the **>>** button
4. Click the **Dependents** tab; then click the **Nasdaq** and __**NYStockExchange** services __in the **Available Services** list and click the **>>** button.
5. Click the **Save** button in the **Edit Service** tab tool bar.

At this point, the Services portlet within Service Navigator portlet should display the Services shown in *Figure 1 Online Services template structure.*

Create Services using the Discovery Library Toolkit

The TBSM V4.2 Discovery Library support includes a method of importing resource information from TADDM or from Discovery Library Books (also known as IdML books). Discovery Library books are produced by a Discovery Library Adapter (DLA) and included with the product. These books provide consistent definitions for the resources the product is managing based on the IBM Common Data Model (CDM).

Imported resource information is then mapped to templates defined in the TBSM software, providing a quick way of building business services in TBSM. The TBSM Service Component Repository (SCR) provides a bridging area between resources and their relationships as represented in CDM and the TBSM business service template model. As CDM resources are moved into the TBSM model, the instance class structure and relationships must fit or be "mapped" into the business model implemented by TBSM. The SCR provides a default implementation of the mappings defined through a set of XML files. This mapping may be customized by editing the XML files. When all the resources are loaded into the SCR, you can create business services using these resources. Create a new service and add the resources from the SCR as dependents and save the service.

In this section, you create TBSM services using a Discovery Library book generated by the ITCAM for SOA DLA. The example book is included as part of the launchDefinitions.zip file. However because the eventual purpose of this example is to launch back to the source product and you do not have access to the ITCAM systems listed in our sample book, you might want to follow the steps indicated here using a DLA book created from a Tivoli product in your own environment. Products that generate an IDML book using a DLA include but are not limited to IBM Tivoli Monitoring, IBM Tivoli Network Manger, and other Tivoli z/OS products.

Import resources using a Discovery Library Book (SOA example)

Create the resources in the SCR by completing the following steps:

1. Install the TBSM Discovery Library Toolkit and ensure that the toolkit service (Windows) or daemon (UNIX®) is running. See the IBM Tivoli Business Service Manager V4.2 installation and administration guides for detailed instructions on how to install and use the toolkit to read discovery data from either TADDM or from Discovery Library books.

2. Run the DLA and produce an IDML book. Each product's DLA has unique instructions on how to create the Discovery Library book so refer to product documentation for details on generating a book. For ITCAM for SOA DLA, the readme.txt has instructions on how to set up the DLA but essentially you run the KD4_DLA command from the DLA_INSTALL_DIR\bin directory.

3. After the book is created, copy it to the computer where TBSM Discovery Library Toolkit is installed and to the directory that you configured the toolkit to read books from. The defaults are:

- For UNIX systems: $TBSM_HOME\discovery\dlbooks

- For Windows® systems: %TBSM_HOME%\discovery\dlbooks
  If the default polling interval set for the toolkit to check for new books is unchanged, then the new book are read in 15 seconds or less. The %TBSM_HOME%\XMLtoolkit\log\msgGTM_XT.log issues a message indicating when the book has been successfully processed.

4. Log in to the Tivoli Integrated Portal console. In the **Task List** on the left side of the console, select **Administration** -> **Service Administration** to open the Service Administration page. The **Service Navigation** portlet on that page shows the **Templates** portlet by default.

5. From the Service Navigation menu, select **Service Component Repository** and then expand the Component Registry. Depending on the DLA book you are using and the type of resources it defines you might need to expand specific nodes (i.e. Servers) to see the service models created. For SOA, expand the SOA Components node to see resources that map to SOA operations.

Create SOAServices Template

To create a template (SOAServices) for the service model that uses the SOA components from the SCR, complete the following procedure:

1. Go back to the **Templates** portlet in the **Service Navigation** portlet and click the **Create New Template** button. The **Edit Template** tab opens in the *service editor*.

2. Type SOAServices __for **Template Name** field.
3. Optionally click the **Display Icon: Browse** button to select the icon you want for SOAServices.
4. Click the **Rules** tab. Then click the **Create Good, Marginal, Bad Aggregation Rule** button . The rule you create in this window specifies that services are dependent upon the child services of the**BSM_SOAOperationPort** template.
5. Enter BSM_SOAOperationPortsWorst for the **Rule Name** field*,* select **BSM_SOAOperationPort** from the menu for the **Child Rule/Mapping** field, and select *any child* for **Condition Type**.
6. Click the **Save** button in the **Edit Template** tab tool bar.

Create AcmeService Service

Now you are ready to create a TBSM service called AcmeService that uses the SOAServices template created in the previous section and the SCR resources created from the SOA DLA book.

1. From the Service Navigation menu, select **Services,** and then click the **Create New Service** button. The **Edit Service** tab opens in the *service editor*.
2. For both the **Service Name** and **Display Name** fields, type AcmeService.
3. In the **Templates** tab, click the *SOAServices* template in the **Available Templates** list and click the **>>** button
4. Click the **Dependents** tab; then click the "Add From Service Component Repository" button. This opens a window where you can expand the nodes in the SCR until you find the service models you want. In this case, expand the SOA Component nodes and click a few services (such as getEnterpriseCustomer, getFromEnterpriseCustomerDB, getFromThirdPartyCustomerInfoDB, and lookupCustomerInNewDB). Then click the **Close** button at the bottom of the window. You should see the services you selected in the **Selected Services** list on the left side of the window.
5. Click the **Save** button in the **Edit Service** tab tool bar.

At this point, the Services portlet within Service Navigator portlet shows the services .

Downloads

launchDefinitions.zip

References

IBM Tivoli Business Service Manager Information Center at *http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/welcome.htm ***includes administration, installation, customization, service configurations guides, and additional related documentation.**

Copyright

Integrating IBM Systems Director With IBM Tivoli Business Service Manager

Integrating Networking with Business Service Management

Integrating Networking with Business Service Management

This paper is a guide to using the new facilities in IBM® Tivoli® Tivoli Network Manager IP Edition (ITNM IP) to more closely integrate networking aspects with business services. It leverages new capabilities of ITNM IP to automatically detect the "edge" of your network, so you can better represent the dependencies business services have on networking infrastructure in both IBM Tivoli Application Discovery Dependency Manager (TADDM) and IBM Tivoli Business Service Manager (TBSM).

In this document, we will guide you through the following:

1. Configuring ITNM IP to detect the network "edge" resources
2. Configuring ITNM IP to display a view of just those resources
3. Exporting the edge resources from ITNM IP, and importing them into TADDM
4. Troubleshooting any naming discrepancies that might arise.

To read this paper, click here.

Integrating TBSM 3.1 and TBSM 4.2 - An ESDA Solution

Overview

The IBM® Tivoli® Business Systems Manager (TBSM 3.1) ESDA feature is designed as an aid for representing TBSM 3.1 Business System folders in an IBM Tivoli Business Service Manager (TBSM 4.2) system. This feature utilizes the External Service Dependency Adapter (ESDA) capabilities of TBSM 4.2 to accomplish this task. The TBSM 4.2 feature incorporates knowledge of the TBSM 3.1 database schema that would not generally be known to a customer trying to develop this capability on their own.

This paper discusses 3 main topics:

- Installing and Configuring the TBSM 3.1 ESDA feature
- Comparing the 3.1 and 4.2 models
- Exploiting TBSM 4.2 - Enhanced Executive Dashboard

Terminology

Table 1 includes definitions of several key concepts from TBSM 3.1 and TBSM 4.2.

Table 1: Key concepts for TBSM 3.1 and TBSM 4.2

Concept	TBSM Version	Definition
External Service Dependency Adapter (ESDA)	4.2	ESDAs gives you the ability to import service models dynamically from an external data source. The model adds and deletes services as required to stay consistent with the external data source.
Data source	4.2	A data source defines connection information for an external database. A variety of SQL based Database Management Systems are supported. This feature includes a data source definition that must be configured to access the TBSM 3.1 database server.
Data fetcher	4.2	A data fetcher polls an external data source to provide event data on a periodic basis. For this feature, a data fetcher is added to poll the TBSM 3.1 database for status.
Business System folder	3.1	A business system folder is a container business system, not linked to any physical resource. It is generally created manually using the TBSM 3.1 Console and represents a business function of the enterprise. Folders can be constructed in a hierarchy to represent how the business is organized.
Business System folder shortcut	3.1	A business system folder shortcut is a folder that is being used in multiple business contexts. These are created in the TBSM 3.1 Console by dragging

		an existing business system folder into another context. The shortcut can have its own properties in 3.1, but often shortcuts are just reusable folders that multiple business systems depend on.
Business System resource	3.1	A business system resource is a business system that represents a specific physical resource. These are created in the TBSM 3.1 Console by dragging a physical resource and dropping it on an existing business system (folder, shortcut, or even another business system resource).

Description

In its default form, the TBSM 3.1 ESDA feature discovers business system folders and creates corresponding services in the TBSM 4.2 system. Dependencies are created in TBSM 4.2 to reproduce the business system folder hierarchy that exists in TBSM 3.1. For business system folder shortcuts, a single 4.2 service is created. The multiple contexts that the shortcuts represent become additional dependencies in version 4.2. This behavior can be modified and this topic is covered in more detail later in this paper.

It is important to note that business system resources are not collected from the TBSM 3.1 system for representation as version 4.2 services. Business system resources represent a broad range of classes of TBSM 3.1 physical resources, and there are potentially large volumes of these resources. This feature is focused only at the business system folder level, and attempts to create a service model that is primarily of interest to executives, business service owners, or business application owners.

To retrieve the TBSM 3.1 service model, a data source is defined by this ESDA feature. It must be configured to access an existing TBSM 3.1 database server. In addition, a data fetcher is added to TBSM 4.2 that periodically gets status data for the business systems from the TBSM 3.1 database and reflects the status on the TBSM 4.2 services.

Users of this package should be familiar with TBSM 3.1. Configuration requires some administrative authority in the TBSM 3.1 Console. Knowledge of the TBSM 4.2 ESDA feature is required to understand how the service model is built from the queries against the TBSM 3.1 database. Finally, familiarity with the TBSM 4.2 Console is needed to take full advantage of the service model created by this feature.

Installing and Configuring the TBSM 3.1 ESDA Feature

This paper is supplemented with a set of attachments that are used to install and configure the TBSM 3.1 ESDA feature. These attachments include the following items:

- A set of files with the extension "update" that is used to set up the optional launch capability for the TBSM 3.1 Web Console
- A set of icons used to represent the TBSM 3.1 business services in the TBSM 4.2 service model
- A set of scripts that can be used with the TBSM 4.2 rad_radshell utility to configure the service model to be created by the TBSM 3.1 ESDA feature

The following information should be noted prior to beginning the installation:

Several TBSM 4.2 environment variables are referenced in this paper, including:

- TBSM_HOME: Home directory for TBSM utilities and other product files
- TBSM_DATA_SERVER_HOME: Home directory for TBSM 4.2 data server
- TBSM_DASHBOARD_SERVER_HOME: Home directory for TBSM 4.2 dashboard server

- TIP_HOME: Home directory for TBSM 4.2 Tivoli Integrated Portal installation

  When referencing the environment variables on Windows® systems, use the syntax %variable%. On UNIX® and Linux® systems use the syntax $variable. This paper uses only the variable name with no variable indicators when no platform context is indicated.

  On Windows systems, the environment variables are set by the installer. On UNIX and Linux systems, you must source TBSM setup scripts to set the environment variables. Switch to the TBSM installation directory (default /opt/IBM/tivoli/tbsm). Run the following commands on the TBSM Data server and Dashboard server, or run both if the servers are on the same system. Note the period mark (.) followed by a space for sourcing the scripts:

- . bin/setupTBSMData.sh **(Data server only)
- . bin/setupTBSMDash.sh **(Dashboard server only) **

  Directory specifications in the installation instructions use the UNIX and Linux style forward slash separator (/). For Windows systems, be sure to replace all forward slashes with the backward slash ().

  Always make a backup copy of any file being updated or replaced by these installation instructions.

Add icons for TBSM 3.1 services

The icon images used for business systems in TBSM 3.1 are used for the services created in TBSM 4.2. The following update only adds new files to the target directories.

1. Switch to TBSM_DASHBOARD_SERVER_HOME/icons directory.
2. From the attachments, copy the TBSM31_bussys_svg.gif. icon file
3. Switch to the svg subdirectory.
4. From the attachments, copy the TBSM31_bussys.svg icon file.

Default configuration

The default configuration creates the templates to be used for TBSM 3.1 services. TBSM31_Model **is the primary template and contains the ESDA rule for retrieving the service instances from the TBSM 3.1 database. The TBSM31_Status template defines a rule for determining the status of a 4.2 service based on the results of a data fetcher that runs against the 3.1 database.

As noted earlier, this default configuration collapses the TBSM 3.1 shortcut folders into a single 4.2 service instance. In addition the default configuration creates the data source and data fetcher used to retrieve data from the TBSM 3.1 database. It also creates one or more seed services, based on the TBSM 3.1 top level business systems that are to be reflected in 4.2.

Preparing for the default configuration

The following information is required to configure the TBSM 3.1 ESDA feature:

- The host name or IP address of the TBSM 3.1 database server system, and the port number being used by the database server, if not the default for Microsoft® SQL Server
- A user ID and password that can access the TBSM 3.1 database
- The host name or IP address of the TBSM 3.1 Console server system, and the port number if the default TBSM 3.1 value is not being used
- The ID, display name, and description of the top level TBSM 3.1 business systems to be added to the TBSM 4.2

service model. The display name and description are available from the TBSM 3.1 Console. An Administrator can turn on the cid:id display in the TBSM 3.1 tooltips function using the Administrator Preferences window. Then the cid:id are available by hovering over a business system with the mouse pointer. The id values are required, while the cid is not required because only LOB resources are retrieved from TBSM 3.1.

Default configuration - Templates, data source, and data fetcher

The first step in the default configuration is to define the templates to be used for TBSM 3.1 based services and the data source and data fetcher used to access the TBSM 3.1 database. Run the following steps to begin the default configuration for the TBSM 3.1 ESDA feature.

1. Edit the tbsm31esda_config.radsh attachment file. Look for all occurrences of the string "CONFIGURATION REQUIRED" and follow the instructions in the comments to configure the settings.
2. From the directory containing the modified attachment file, run the following command:
   Windows systems: type tbsm31esda_config.radsh | %TBSM_HOME%\bin\rad_radshell.bat

   UNIX and Linux systems: cat tbsm31esda_config.radsh | $TBSM_HOME/bin/rad_radshell
3. See 2.3.2, "Create services for TBSM 3.1 business system folder shortcuts", for an alternate configuration for the template.
   Important: The template configurations are mutually exclusive, so only one can be applied to a single TBSM 4.2 system. To switch configurations, first remove any services and templates created with the previous configuration. See 2.4, "Removing the templates, services, data source, and data fetcher", for more information.

   Verify the results by looking for the following in the Service Navigation portlet of the TBSM 4.2 Service Administration page:

- Templates navigation view: templates named **TBSM31_Model** and **TBSM31_Status**
- Data navigation view: a data source named **TBSM**
- Data Fetcher navigation view: a data fetcher named **TBSM_AlertStatus**
- If the expected results are not seen, see section 5 "Troubleshooting".

Tips for using ESDAs in TBSM 4.2

The behavior of services with ESDA rules is controlled by a special set of properties. ESDA services must become "invalidated" in order for the ESDA rules to run. Even then ESDA rules are only run when a service is expanded in the tree view or when a canvas view requires the dependencies of a service in order to show the complete view.

By default, the services created by running the ESDA rules included in this feature are automatically invalidated every 60 seconds. This means that there is minimal delay in detecting when business system folders have been added and deleted in TBSM 3.1. However, it means more execution of the SQL queries in the ESDA rules, and thus more overhead for the TBSM 4.2 server.

Important: The services tree in TBSM 4.2 does not automatically add and remove services from nodes that are expanded or have been expanded. One of these nodes could be a TBSM 3.1 based service that has been invalidated and will be adding or removing services when the ESDA rule is next run. It is necessary to refresh the 4.2 services tree and click the expand button again to drive the ESDA rules for such an "expanded" node.

See the TBSM 4.2 *Service Configuration Guide* for detailed information about working with ESDAs.

Tips for using data fetchers in TBSM 4.2

Data fetchers have various properties that can be set to tune the performance and to make the behavior more like the standard event processing behavior for the Omnibus events. By default, each data fetcher caches 100 rows retrieved by the last run of the fetcher. This number can be set to more closely match the amount of rows actually being retrieved.

On the next fetch of the data, the results are compared to the cached results and only changed rows are processed. The idea is to limit the number of rows that must be compared to the incoming status rules that use the fetcher across all the templates.

This behavior is not well suited for this feature, as the services created by ESDA rules might not yet exist when the data fetcher starts running. Thus the data fetcher in this feature uses a filtering expression to ensure all rows of data are processed on each call of the data fetcher, keeping the status from TBSM 3.1 as current as possible. This disables the default caching that typically occurs for a TBSM 4.2 data fetcher.

The data fetcher is coded as efficiently as possible, but will likely retrieve more business systems instances from TBSM 3.1 than there are corresponding services in TBSM 4.2. A small amount of data is retrieved for each row, and only one rule is defined to use the data fetcher, so the performance should not be adversely affected.
See the TBSM 4.2 *Service Configuration Guide* for detailed information about working with data fetchers.

Default configuration: Seed services

The second step in the default configuration is to define the seed services to be created in TBSM 4.2. These services correspond to the top level TBSM 3.1 business systems that you want to display in the TBSM 4.2 service model. Run the following steps to complete the default configuration for the TBSM 3.1 ESDA feature:

1. Edit the tbsm31esda_selected_seeds.radsh attachment file. Look for the string "CONFIGURATION REQUIRED" and follow the instructions in the comments.
   Select only top level TBSM 3.1 business systems that are to be represented in TBSM 4.2. The id and the display name are required for the TBSM 3.1 business system (LOB) folders to be retrieved. The TBSM 3.1 description can be used, or any preferred text can be specified. The business system folders specified in this file *must not* be business system folder shortcuts or business system resources.
   Important: It can be beneficial to start with a single business system folder to verify that the feature is installed and configured correctly. Later, you can run the script as many times as necessary to add additional top level business systems.

2. From the directory containing the modified attachment file, run the following command:
   Windows systems: type tbsm31esda_selected_seeds.radsh | %TBSM_HOME%\bin\rad_radshell.bat

   UNIX and Linux systems: cat tbsm31esda_selected_seeds.radsh | $TBSM_HOME/bin/rad_radshell

3. See 2.3.3, "Include all top level business system folders", for an alternate configuration for the seed services.
   Important: The seed configurations are mutually exclusive, so only one can be applied to a single TBSM 4.2 system. To switch configurations, first remove any services created with the previous configuration. See 2.4, "Removing the templates, services, data source, and data fetcher", for more information.

   Using the TBSM 4.2 Console, open the Service Administration page and select the Services navigation view in the Service Navigation portlet. Verify that the seed services that were specified in the configuration are displayed at the root level of the service tree.

   If the expected results are not displayed, see section 5 "Troubleshooting".

Optional Configurations

The following sections document the optional configurations that can be performed for the TBSM 3.1 ESDA feature, including the following configurations:

- Launching the TBSM 3.1 Web Console
- Create services for TBSM 3.1 business system folder shortcuts
- Include all top level business system folders

Launching the TBSM 3.1 Web Console

This optional feature adds an action and right-click menu option for launching the TBSM 3.1 Web Console. The cid/id information, as well as the target console server, is configured with each TBSM 3.1 service that is represented in TBSM 4.2. The TBSM 3.1 Business Impact view is launched by default.

The menu item is available as a submenu item of the **Integrations** menu and will only be enabled for TBSM 3.1 services collected by the ESDA feature. The first launch to the Web Console requires you to log in. If the browser with the Web Console is left open, each subsequent launch uses the same Web Console session and adds additional views.

Important: The following configuration steps can touch several TBSM 4.2 product files. Be sure to make backup copies of the files prior to merging the changes for the launch capability.

To add the Launch capability, complete the following steps:

1. This first step makes the action available for use by TBSM 4.2 menus. Switch to the TBSM_DATA_SERVER_HOME/av/xmlconfig directory. Edit the canvasOpenURLActions.xml file and merge the changes from the canvasOpenURLActions.xml.update attachment file as follows:
1. Locate the </canvasConfig> tag: This is the matching end tag for <canvasConfig> and should be at or near the bottom of the file.
2. Copy the content of the canvasOpenURLActions.xml.update file immediately above the </canvasConfig> tag. The openURLAction with name="LaunchTBSM31" should now be the last action before the </canvasConfig> tag.
2. The action will now be added to the **Launch to** submenu in TBSM 4.2. From TBSM_DATA_SERVER_HOME/av /xmlconfig, edit the canvasDynamicSubMenuActions.xml file. Merge the changes from thecanvasDynamicSubMenuActions.xml.update attachment file as follows:
1. Locate the <dynamicSubMenuAction> tag with the "name" attribute set to "IntegrationTools".
2. Locate the matching end tag </dynamicSubMenuAction>.
3. Copy the content of the canvasDynamicSubMenuActions.xml.update attachment file immediately above the end tag located in step b.
3. This step makes the attributes required to enable the menu item available to the view definitions. Only the following views should be updated, and only if you want to launch the TBSM 3.1 Web Console from the view. For any view definition not updated, the menu item is displayed, but is always disabled.

- ViewDefinition_BasicRelationships.xml
- ViewDefinition_BusinessImpact.xml
- ViewDefinition_BusinessImpactAll.xml
- ViewDefinition_Concentric.xml
- ViewDefinition_CustomView.xml
- ViewDefinition_GIS.xml

- ViewDefinition_Grid.xml
- ViewDefinition_Relationships.xml
  From TBSM_DATA_SERVER_HOME/av/xmlconfig, edit each view that will support the launch. Merge the changes from the

  ViewDefinition_.xml.update attachment file as follows:

1. Locate the </dataTypeMapping> tag: This is the matching end tag for <dataTypeMapping>.
2. Copy the content of the ViewDefinition_.xml.update attachment file immediately above the </dataTypeMapping> tag. The new cid, id, consoleServer, and consoleServerPort fields should now be included in the dataTypeMapping definition.

4. Restart the TBSM 4.2 Data server to pick up the modifications:

- Windows systems:

1. From Administrative Tools, open the Services window.

2. Right-click "Tivoli Business Service Manager - TBSMProfile_Port_17310", and select Restart.

- UNIX and Linux systems:

1. cd to $TIP_HOME/profiles/TBSMProfile/bin
2. Enter ./stopServer.sh server1 -username *tipadminuser* -password _tipadminpassword
   where _tipadminuser and *tipadminpassword* must be replaced with the Tivoli Integrated Portal administrative user and password for the TBSM 4.2 configuration

5. Enter ./startServer.sh server1

   To verify that the launch action has been added do the following steps:

1. From the TBSM 4.2 Console, open the Service Administration page and select the Services navigation view in the Service Navigation portlet.
2. Click on a service created from a TBSM 3.1 business system.
3. In the **View Service** tab that displays the canvas view, right-click one of the services displayed in the canvas view.
4. Verify that the **Launch to** menu includes a submenu item called **Launch TBSM 3.1.**
5. If the menu item is enabled, click it to verify that the TBSM 3.1 Web Console is launched.
6. If the menu item is not shown, not enabled, or does not launch the TBSM 3.1 Web Console, see section 5 "Troubleshooting".

Create services for TBSM 3.1 business system folder shortcuts

This optional configuration should be used when business system folder shortcuts are to be represented in TBSM 4.2 as distinct services. This might be necessary if shortcut folders in TBSM 3.1 have unique properties, especially properties that can affect the status. This includes, for example, Percentage Based Threshold (PBT) rules and instance level thresholds. The assumption is that the business system behaves differently when being managed in multiple contexts through the use of shortcuts.

In this configuration, when a shortcut is encountered, a distinct 4.2 service is created, with an attribute that defines the id of the source business system folder in TBSM 3.1. When this newly created service retrieves its child business systems, the source id is used, because in TBSM 3.1, only the source business system folder has links to child business systems.

Important: This configuration must not be applied to a system that already has the default configuration defined (see 2.2.2, "Default configuration - Templates, data source, and data fetcher"). The same template names are used and will result in conflicts when trying to import this configuration. Remove the TBSM 3.1 templates and any services created by this ESDA feature before updating the configuration to allow shortcuts. See 2.4, "Removing the templates, services, data source, and data fetcher", for more information.

Preparation for this configuration is the same as for the default configuration. See 2.2.1, "Preparing for the default configuration", to make sure that all the required information has been gathered before you begin.

Run the following steps to configure the TBSM 3.1 ESDA feature to support shortcuts:

1. Edit the tbsm31esda_config_shortcuts.radsh attachment file. Look for all occurrences of the string "CONFIGURATION REQUIRED" and follow the instructions in the comments to configure the settings.
2. From the directory containing the modified attachment file, execute the following command:
   Windows systems: type tbsm31esda_config_shortcuts.radsh | %TBSM_HOME%\bin\rad_radshell.bat

   UNIX and Linux systems: cat tbsm31esda_config_shortcuts.radsh | $TBSM_HOME/bin/rad_radshell

   Verify the results by looking for the following items in the Service Navigation portlet of the TBSM 4.2 Service Administration page:

- Templates navigation view: Templates named **TBSM31_Model and TBSM31_Status**
- Data navigation view: A data source named **TBSM**
- Data Fetcher navigation view: A data fetcher named **TBSM_AlertStatus**

If the expected results are not displayed, see section 5 "Troubleshooting".

Include all top level business system folders

This configuration defines a single seed service that is the root of the TBSM 3.1 service model. All TBSM 3.1 top level services are automatically discovered by running the ESDA rule against this top level seed. This single seed is called **TBSM31_Systems** by default.

Important: This configuration should not be applied to a system that is already using selected TBSM 3.1 services, which is the default implementation (see 2.2.3, "Default configuration - Seed services"). Remove any previously defined seed services before proceeding. See 2.4, "Removing the templates, services, data source, and data fetcher" for more information.

Run the following steps to configure the TBSM 3.1 ESDA feature to automatically retrieve all top level business system folders:

1. Edit the tbsm31esda_single_seed.radsh attachment file. No configuration is required unless you want to change the display name for the single seed service.
2. From the directory containing the modified attachment file, run the following command:
   Windows systems: type tbsm31esda_single_seed.radsh | %TBSM_HOME%\bin\rad_radshell.bat

   UNIX and Linux systems: cat tbsm31esda_single_seed.radsh | $TBSM_HOME/bin/rad_radshell

   If the expected results are not displayed, see section 5 "Troubleshooting".

Removing the templates, services, data source, and data fetcher

If the default configuration has been installed for the templates, and is not satisfactory, the templates, services, data source, and data fetcher that are created must be removed before changing the configuration. This would

also be the case if you are switching from an optional configuration back to the default configuration.
If the only configuration change is to modify which seed services will be used, then only the service instances need to be removed.

The following steps outline how to remove the services, templates, data fetcher, and data source added by this feature:

Log in to the TBSM 4.2 console and select the Service Administration page for all the following tasks.

Service instances:

1. In the Service Navigation portlet, select the **Services** navigation view.
2. Click the **Delete** button ( ) to display the Delete Instances page.
3. Select all instances with names of the form **TBSM31_LOB:nn** and click the **Delete** button ( ).

Important: If the single top level seed is no longer needed, delete the instance named **TBSM31_Systems** as well.

The following items should only be deleted if the configuration is being changed from the default, collapsed shortcuts, to the optional, supporting shortcuts, or vice versa. If the only change is the configuration for the seed services, these items should be retained.

Templates:

1. In the Service Navigation portlet, select the **Templates** navigation **view.
2. Click the **Delete** ( ) button to display the Delete Templates page.
3. Select the **TBSM31_Model** and **TBSM31_Status** templates and click the Delete button ( ).

Data fetcher:

1. In the Service Navigation portlet, select the **Data Fetcher** navigation view.
2. Right-click the row containing the TBSM_AlertStatus data fetcher ***and select *Delete** from the context menu.

Data source:

1. In the Service Navigation portlet, select the **Data** navigation view.
2. Right-click the row containing the TBSM data source and select **Delete** from the Context menu.

Comparing the TBSM 3.1 and 4.2 models

This section provides more detail on how the TBSM 3.1 business systems are represented in TBSM 4.2. Several screen captures are used to show the differences between the two models. While some instructions might be included in this paper about how to view the model in 3.1 and 4.2, it is assumed that the reader is familiar with the consoles provided with each product.

For more information on using the TBSM 3.1 Console, refer to TBSM 3.1 publication *Introducing the Consoles*

For more information on using the TBSM 4.2 Console, refer to the following TBSM 4.2 publications:

- *Service Configuration Guide*
- *Customization Guide*

Business Systems tree → Service Model hierarchy

Figure 2: TBSM 3.1 Business Systems tree

See Figure 2 for an example of a TBSM 3.1 Business Systems tree. The following list defines the resources shown in the tree.

Business Systems: This is the root of the Business Systems tree. It does not show any status and is not reflected as a service in the default configuration for 4.2.

Email, InternetBanking, Payroll: These are the top level business systems. These are the candidates to be configured for the 3.1 ESDA feature as top level services in TBSM 4.2.

DBServers, NetworkServers: These are child business systems of **Email**. Because these business systems are both folders, they are automatically added as dependent services of **Email** in the 4.2 service model created when using this feature.

OS001, OS002, OS004, OS005: These are business system resources. Services are *not* created for these business systems in TBSM 4.2. These represent TBSM 3.1 physical resources, as indicated by the icon, which is the TBSM 3.1 icon used for an operating system.

MyResources - admin: This is a TBSM 3.1 Critical Resource List (CRL) for user "admin". While CRLs are business system folders, they are not valid for configuration as top level services. If the optional configuration is used to include all top level business systems (see 2.3.3, "Include all top level business system folders"), the CRLs are skipped automatically.

Figure 3 shows the same business systems, now as services in a TBSM 4.2 service model. The default configuration of the ESDA feature has been applied in this case, with business systems **Email**, **InternetBanking**, and **Payroll** configured as seed services.

The icon for business system folders has been carried forward to TBSM 4.2 to help identify services based on TBSM 3.1 business systems. Notice only the folders and child folders are present and there are no business system resources.

The topology view from TBSM 3.1 provides similar capability to the TBSM 4.2 canvas view. Figure 4 shows a side-by-side comparison of the 3.1 and 4.2 models in graphical form.

Business system folder shortcuts

The most interesting decision to be made when configuring the TBSM 3.1 ESDA feature for TBSM 4.2 is in regards to the treatment of business system folder shortcuts. Shortcuts are intended for a TBSM 3.1 customer to define a business system that is reusable as a dependency in other business contexts. TBSM 3.1 takes it one step further and allows these shortcuts to have distinct properties, most notably the threshold settings and Percentage Based Threshold (PBT) rules that can affect propagation. By default, the shortcuts have the same properties as the source business system from which they were created.

TBSM 3.1 has the following method of handling shortcuts:

- A shortcut has a link to the business system from which it was created.
- Shortcuts have no unique child business systems. The link to the source business system is used to determine child business systems. A source business system and all of its related shortcuts always have the same child business systems.
- By default, a shortcut inherits all properties from its source business system. These are modifiable per shortcut, allowing, for example, different propagation thresholds for a shortcut.

Default configuration - shortcuts are collapsed

By default, this TBSM 3.1 ESDA feature simplifies the handling of shortcuts by only creating a 4.2 service for the non-shortcut business systems. When a shortcut is encountered by the ESDA rules, a 4.2 dependency is created from the parent service to the service that was created for the source business system. Thus, a business system with two shortcuts in TBSM 3.1 becomes a service in 4.2 that has three parent services, one for the source business system, and one for each shortcut.
Consider the TBSM 3.1 **Payroll** business system shown in Figure 5.

The **CheckPrinting** business system was originally defined as a child business system of the **California** business system folder. *****The business system was then "reused" under *NewYork** and**NorthCarolina**, resulting in two business system folder shortcuts (notice the icon that denotes a shortcut:).

Figure 6 shows the resulting service model in TBSM 4.2. Notice that the tree looks the same, but the canvas view tells what is really happening; there is one service with three parents.

Optional configuration - create 4.2 services for shortcuts

Using the optional configuration described in 2.3.2, "Create services for TBSM 3.1 business system folder shortcuts", business system folder shortcuts can be created as distinct services in TBSM 4.2. With this configuration, the unique status of shortcuts will be reflected in 4.2. On the downside, duplicate display names are likely to appear in the 4.2 service model, with no visual designation of which services represent shortcuts. However, after the shortcut becomes a service in 4.2, it really no longer matters that it was a shortcut in 3.1, as no special behavior is attached to it in 4.2.

Important: Internally, the ESDA rule recognizes by a separate id attribute that the service was a 3.1 shortcut. This is necessary in order to retrieve the dependencies for the service from 3.1, but the resulting model shows no evidence of this internal handling of the "shortcut" service.

The default configuration is designed with the assumption that the shortcuts reflect only reuse of the business system, without the complexity of configuring each shortcut to have different properties.

In the example illustrated by Figure 7, the **CheckPrinting** business system has been defined to propagate status differently for each context in which it is being used. With the default configuration of this TBSM 3.1 ESDA feature, you see all the contexts (parents) for **CheckPrinting**, but only the status for the non-shortcut copy of **CheckPrinting** will ever be shown in the 4.2 service model. Thus a Red status is not reflected in TBSM 4.2 for this case.

By using the optional configuration for handling shortcuts, the 4.2 service model can look very much like the 3.1 model.

Business Impact

TBSM 3.1 supports a Business Impact view that is designed to show operators how a particular outage is affecting the overall business. The basic design is a "bottom up" view starting from a resource or business system and traversing up the hierarchy of ancestors to show how the starting resource affects different business systems. This is one of the signature functions of TBSM 3.1.

The TBSM 4.2 Business Impact view starts from a selected service and displays a "bottom up" relationship view, thus showing how the service impacts the services above it in the service model. Only services that are*not* in a normal state are shown in the view. An alternative "Business Impact All" view shows all services impacted by the starting service, including services that are in a normal state.

It is important to note that the TBSM 3.1 Business Impact view content is determined by a database stored procedure. It is designed to handle physical resources as well as business systems, and there is special logic to deal with business system resources and business system folder shortcuts.

With business system resources not a consideration for this TBSM 4.2 feature, the 4.2 Business Impact view is much simpler. It is a customized version of the 4.2 BasicRelationships view that specifies the view be built by looking only "up" through the dependencies hierarchy. The number of levels is set artificially high to try and ensure that the full range of business impact is included.

Figure 10 shows an example of a TBSM 3.1 Business Impact view. It is displayed using the TBSM 3.1 "HyperView", which is a graphical presentation designed to show large numbers of objects. It has been filtered to show resources with a status of Yellow or worse.

The TBSM 4.2 Business Impact view, shown in Figure 11, is a similar view. By default, because shortcuts have been collapsed, the view is a bit simpler. Also note that the 4.2 view is designed to automatically hide any services that are Green, thus showing only the degraded services.

Important: A Business Impact view that has only Green services in TBSM 4.2 will be empty.

Exploiting TBSM 4.2 - Enhanced Executive Dashboard

Now that the installation and configuration of the TBSM 3.1 ESDA feature is complete, and the resulting 4.2 service model has been explored, it is time to talk about exploiting the many features of TBSM 4.2 with the new services.

Because the focus of this feature is to bring high level TBSM 3.1 business system folders into the 4.2 service model, the target audience is business and IT executives, as well as business system and application owners. TBSM 3.1 introduced an Executive Dashboard feature, along with an Application Programming Interface that customers can use to retrieve executive service data for their own Web applications.

Using the capabilities of TBSM 4.2, an enriched Executive Dashboard can be created. It can be highly customized to the needs of a specific enterprise. There are two main areas of improvement in 4.2 that allow this to be accomplished:

Data integration:

- Built-in Service Level Agreement (SLA) data
- Integration with external data sources

Visualization:

- Customizable canvas views
- Customizable scorecards

Preparation

The first steps for building the customized Internet Banking page are defining templates to be used for the data integration rules and defining the executive user that views the instances associated with these templates.

Defining the data integration templates

For the example Internet Banking page, rules will be created to process the external data that measures the health of our fictional Internet Banking service model. The rules are different for the lower level services than the containing parent services, and thus two templates are required.

For this example, create the following two templates:

TBSM31_InternetBankingChild: This template will have rules to process external events and calculate attributes indicating the health of the dependent services

TBSM31_InternetBankingParent:This template will have rules that aggregate the values from the child template rules to provide overall health information for the high level service

For now, provide only the name and, optionally, a description. The icon is not really relevant, as these templates are not primary templates in this example. The templates just need to exist at this point so that the viewing privileges can be assigned for the executive user that will be defined in the next section.

Defining the executive user

The next step in building the Executive Dashboard for TBSM 4.2 is to define the executive user. A TBSM 4.2 user with the role "iscadmins" is required to perform this task. After it is created, this executive user is given the authority to view only service instances relevant to this executive's portion of the business. Customized Web pages, like the one shown in Figure 13, can be created specifically for access by this new executive user.

The following list outlines the steps for adding the new executive user **ibank**:

1. Log in to the TBSM 4.2 console with a user ID that has the "iscadmins" role.
2. In the task list on the left side of the TBSM 4.2 Console, expand the **Users and Groups** folder and select **Manage Users**.
3. In the **Search for Users** pane, select the **Create** button.
4. In the User ID field, enter ibank.
5. Complete the other fields with the values you want.
6. To complete creating the user **ibank,** select the **Create** button.
7. From the **Users and Groups** folder, select **Administrative User Roles**.
8. Select the **Add** button, and type ibank into the User field.
9. Press Ctrl and click the **tbsmReadOnlyUser** and **ncw_admin** roles.
10. To save the new user, select **OK**, and then select **Save**.

Refer to the TBSM 4.2 *Administrator's Guide* for more information on maintaining users, groups, and roles.

The user **ibank** must now be given the privilege to view only the instances related to Internet Banking. This limits the contents of the service tree that can be viewed by **ibank** to the relevant services, which ultimately defines the content of the customized Internet Banking dashboard. See Figure 14 for an illustration of using the **Security** tab of the Template editor page to assign the **RAD - View Instance** privilege to all instances of the child template for the Internet Banking example. Repeat this same security assignment for the parent template.

The privileges can also be assigned on a per-service instance basis. This example assumes the child and parent templates are only assigned to services that are related to Internet Banking. This simplifies the process and also allows the appropriate security privilege to automatically be assigned to new service instances when they are tagged with the Internet Banking child or parent template.

Data Integration

TBSM 4.2 has data integration capabilities that were not available in TBSM 3.1. Built into the TBSM 4.2 product is the ability to access and display data, like Key Performance Indicators (KPIs) and Key Quality Indicators (KQIs), from external data sources. Rules can be created to process this external event data and assign status and attributes to the services that were created with this ESDA feature.

In addition, Service Level Agreement rules can be assigned to the templates associated with the services that were created with this ESDA feature. Refer to the TBSM 4.2 *Service Configuration Guide* for details on how to use the SLA features of TBSM 4.2. This paper does not cover SLA in any more detail, but the standard SLA attributes can be plugged into the same custom views and scorecards that are described herein.

The focus of this paper is the integration of external data into the TBSM 4.2 product to enrich the views and scorecards that can be created. In the following example, there is a fictional database that contains Key Performance Indicators for the business services in this fictional service model. Table 2, "Example KPI data for integration with TBSM 4.2 services", shows the layout of the example KPIData SQL table.

Table 2: Example KPI data for integration with TBSM 4.2 services

See the Appendix for SQL code that can be used to create the sample table and sample data used by the Internet Banking example.

Adding the data source and data fetcher

The first step in integrating with an external data source is to define it for use by TBSM 4.2. In this example, a data source called **KPI** is created.

1. Log in to the TBSM 4.2 console with a user ID that has administration roles. This should be any user that you added to the TBSM 4.2 group **tbsmAdmins**.
2. From the task list on the left side of the TBSM 4.2 Console, expand the **Administration** folder and select **Service Administration**.
3. From the Service Navigation portlet, select the **Data** navigation view.
4. To create a new data source, select the **Create** icon ( ) .
5. See Figure 15 for an example of how to fill in the data source information.
6. Test the connection to the data source and make any necessary corrections.
7. Save the data source.

You now define one or more data fetchers that use the data source. In this example, a data fetcher called **KPI_InternetBanking** is created.

1. Log in to the TBSM 4.2 console with a user ID that has administration roles. This should be any user that you added to the TBSM 4.2 group **tbsmAdmins**.
2. From the task list on the left side of the TBSM 4.2 Console, expand the **Administration** folder and select **Service Administration**.
3. From the Service Navigation portlet, select the **Data Fetcher** navigation view.
4. To create a new data fetcher, select the **Create** icon ( ).
5. See Figure 16 for an example of how to fill in the data fetcher information. In this case the fetcher gets all the columns from the KPIData table. This fetcher polls at least every 2 minutes and waits no longer than 5 minutes between fetches.
6. To get a preview of the KPI data that will be fetched, select the **View Data** button.

7. Save the data fetcher.

   See the TBSM 4.2 *Service Configuration Guide* for detailed information on adding data sources and data fetchers.

Adding rules to process data fetcher events

The integration of the KPI data with the TBSM 4.2 service instances is completed by creating various rules that are applied when the data fetcher polls and finds new event data. In the Internet Banking example, the rules will match based on the name of the service. After each applicable rule is applied, the affected services contain attributes corresponding to the rule names, with values based on the rule calculation.

The rules will exist on either a "parent" template or a "child" template, with the parent rules aggregating the values from the child templates. In the Internet Banking example, shown in Figure 13, service**InternetBanking** is the parent service, while the child services are the regions designated by **Northeast**, **Northwest**, **Southeast**, *__and *Southwest__.

This paper does not attempt to explain all aspects of defining rules in TBSM 4.2. See the TBSM 4.2 *Service Configuration Guide* for detailed information on creating rules.

Table 3 contains sample rule definitions that were created for the child template in order to produce the Internet Banking executive scorecard.

Table 3: Template rules defined for child services of InternetBanking

Rule name/Attribute name	**Rule type**	**Description**
NewAccounts ClosedAccounts Transactions DownTimeMinutes ActiveAccounts	Incoming Status Rules based on numeric value and using **KPI_InternetBanking** data fetcher	The *servicename* column is used for matching to the correct 4.2 service. Each rule has column *kpitype* set as a filter field. For example,*kpitype* must equal "New" for rule NewAccounts to be applicable. Column *kpivalue* contains the value to be assigned to the rule.
AccountTrend	Numerical formula rule	Calculates the current trend for open versus closed accounts. Formula: NewAccounts.Value - ClosedAccounts.Value
CostOfDownTime	Numerical formula rule	Calculates the cost of the down time. Formula: DownTimeMinutes.Value * 100
Health	Numerical formula rule	Sets a value, using a policy that indicates the health of the service. For details of the policy implementation, see the Appendix.

Note that many of these rules in Table 3 permit the setting of the overall status of a service based on user-defined thresholds for each rule output value. This provides greater flexibility in choosing what metrics actually determine the status for a service.

Table 4 contains sample rule definitions that were created for the parent template in order to produce the Internet Banking executive scorecard. Most of these just aggregate the attribute values for the rules defined in the child template.

Table 4: Template rules for parent service InternetBanking

Rule name/Attribute name	**Rule type**	**Description**
NewAccounts ClosedAccounts Transactions DownTimeMinutes ActiveAccounts	Numerical aggregation rule, each based on the child template described in Table 3.	Each rule is defined using the **Sum** aggregation function. The sum is calculated across the values for the child template rule of the same name.
AccountTrend	Numerical aggregation rule	The sum of the AccountTrend attribute values from the child template.
CostOfDownTime	Numerical formula rule	Calculates the cost of the down time. Formula: DownTimeMinutes.Value * 100
Health	Numerical aggregation rule	The aggregation function is set to the **Maximum** value for the Health attribute from the child template.

Notes on assigning templates and setting identification fields

TBSM 4.2 allows for multiple templates to be assigned to a service instance, with a single template defined as the primary. When assigning the templates containing the rules in Table 3 and Table 4, be sure not to change the primary template. The primary template is defined by the ESDA feature and must not be modified. Figure 17 is a screen capture of the correct Template assignment for the **InternetBanking** parent service. __

The identification fields must also be set properly for Incoming Status rules to be applied to the correct resources. The TBSM 3.1 business systems become services with instance names based on the TBSM 3.1 id value. This is the only means for ensuring uniqueness, because TBSM 3.1 does not enforce unique names for business systems.

For external data sources, like the example KPIData table, the service name is likely the key identifier. TBSM 4.2 supports the correlation of the rules based on varying values by the setting of the Identification Fields. In Figure 18, the top image shows the default settings for the child template rule matching, while the bottom image shows the modifications that are needed to make the rules match based on service name. Note that the identification field for the incoming status rule
TBSM31IncomingStatusRule1 on the **TBSM31_Status** template remains unchanged.

Visualization

TBSM 4.2 supports the following highly customizable means of visualizing the status and KPI data:

- Custom view definitions

- Custom static canvases
- Custom service trees

  Each of these is covered in detail in the following TBSM 4.2 publications:

- *Service Configuration Guide*
- *Customization Guide*

  The following sections give an overview of how these visualization techniques could be used with the Internet Banking example.

Custom view definitions

TBSM 4.2 comes with a default set of view definitions. Using one of these default view definitions, you can build a customized view to show, for example, details of the KPI data being fetched from an external data source. You can also change the behavior of the view definition, deciding, for example how many levels up and down to show in the view, and whether or not to display certain templates.

The following basic outline shows how to create this custom view definition:

1. Open the default **BasicRelationships** view for one of the TBSM 3.1 services that has been added to the 4.2 system; **InternetBanking** in this example.
2. Edit the view definition by selecting the **Edit View Definition** button ( ) You must be in the "full" canvas client in order to edit views. See the TBSM 4.2 *Service Configuration Guide* for more information.
3. Click the **Save As New** button and provide a name for the custom view.
4. Select the **Edit View Definition** button ( ) again to edit the new view. Switch to the **Visuals** tab to modify how the services are to be displayed.
5. From the Service Template drop-down list, select **TBSM31_Model**, which is one of the templates defined by the TBSM 3.1 ESDA feature. It is also the primary template, which dictates which services are affected by the visual customization.
6. From the Type drop-down list, select **FiveElementPrototype** in order to match the illustration in Figure 19.
7. See Figure 20 for how the values are set to build the view that is illustrated in Figure 19_._ Note the use of the syntax @AttributeName. This is required to assign an attribute that is defined by a template rule that is not from the primary template; the child or parent template in this case. The drop-down attribute list only shows attributes from the primary template. See Table 3 and Table 4 for the attributes (rule names) used in this example.

   Important: The cache for the KPI data fetcher needs to be cleared for the new view definition to display the KPI data. Because the data fetcher was already running before the templates were assigned, the services are not updated with the KPI data unless the cache is cleared. This is because TBSM 4.2 only processes changed data rows, by default. It might be better to disable the data fetcher and perform the fetches manually while working with the Internet Banking example. See the TBSM 4.2 *Service Configuration Guide* for more information on working with data fetchers.

Custom static canvases

A custom static canvas is designed to show a fixed set of visual elements with no dynamic change in content. Contrast that with the custom view definition described in the previous section, which applies visual customization to a view that is created dynamically, based on a starting service and its dependencies.

With a custom static canvas, specific service instances are selected to be displayed in the view. Status and other attribute values that are shown are kept up-to-date, but no other service instances are dynamically added or removed from the view.

In addition to the visual elements (known as Indicators) that represent the service instances, other decorations can be added to a custom static canvas, including these items:

- Text fields
- Graphics, for example a logo. User defined graphics can be added to the selection palette.
- Boxes and connecting lines
- Background color

The following basic outline shows how to create the custom static canvas illustrated in Figure 21:

1. From the task list on the left side of the TBSM 4.2 Console, expand the **Administration** folder and select **Service Administration**.
2. From the Service Navigation portlet, select the **Custom Canvases** navigation view.
3. Select the Create button ( ) to create a new custom static canvas.
4. Select the **Indicators** tab. Size the window so that all the indicators are displayed.
5. Select the speedometer style gauge and click on the blank canvas to position the gauge.
6. A wizard is displayed to walk you through the configuration of the indicator. First, select the **InternetBanking** service instance. On the second page of the wizard, set the values as indicated in Figure 22. Note the use of the syntax *@AttributeName*. This is required to assign an attribute that is defined by a template rule that is not from the primary template; the child or parent template in this case. The drop-down attribute list only displays attributes from the primary template .
7. To complete the settings for the gauge, click the **Finish** button.
8. Click the **Decoration** tab to complete adding the logo and the text field. Refer to the TBSM 4.2 *Service Configuration Guide* for more information on using the Inspector tool to further customize the visual elements. For this example, the inspector is required to set the "Active Accounts" text that is displayed in the custom static canvas shown in Figure 21.
9. To save the view, click the Save button ( ) on the toolbar. When prompted, enter a name, for example **ActiveAccounts**.

Refer to the TBSM 4.2 *Service Configuration Guide* for detailed information on creating custom static canvases.

Custom service trees

TBSM 4.2 also allows for the creation of customized service trees by using the Tree Template editor. These trees can show numeric values, text values, or graphics that represent the numeric values. The display of the tree is controlled by TBSM 4.2 policy code, which can also be customized to perform additional formatting for the values in the tree.

Tree Template Editor

In the beginning of "Exploiting TBSM 4.2 - Enhanced Executive Dashboard", a customized Web page was shown as the goal for an Executive Dashboard (see Figure 13). The top half of this page shows a customized service tree (also known as a customized scorecard), which requires a customized tree template. The following steps show how this customized tree template was created:

1. From the task list on the left side of the TBSM 4.2 Console, expand the **Administration** folder and select **Service Administration**.
2. From the Service Navigation portlet, select the **Services** navigation view.
3. Select the **Tree Template Editor button** ( ) to access the Tree Template editor.
4. To the left of the "Tree Template Name" label, select the Create button ( ) to add a new tree template. Enter the name InternetBankingTree to match the example used in this paper.
5. After the window refreshes, select the new tree template name from the drop-down list.
6. See the TBSM 4.2 *Service Configuration Guide* for complete details on how to use the tree template editor. Figure 23 illustrates the general layout of the editor window. Highlighted areas include the custom column names, the templates from which attributes are obtained, and the assignment of attributes to columns.
7. Be sure to set the display attributes for both the parent and child templates by selecting each template in the Active Template list.

Tree Template policy

To realize the full capability of the customized service trees, it might be necessary to update the policy that is run when the service tree is displayed. This policy controls the basic display behavior for the columns in the tree. For example, you can substitute the green, yellow, and red indicators based on a numeric attribute displayed in the tree. Another default behavior is to display a percent sign (%) in columns where the name includes a percent sign.

In the target tree template shown in Figure 13, there are several customizations to the policy that are based on the column names and the numeric values, including these items:

- Negative values are displayed in red in the **Trend** column
- The **Transactions** column changes large numeric values to be expressed with the M and K suffixes
- The **Cost** column displays the values with a preceding dollar sign ($)
- The **Health** column uses the green, yellow, and red indicators

See the Appendix for a listing of the policy that was modified to display the custom service tree. The policy can be edited using the **Edit Policy** button found on the Tree Template editor that is described in the previous section. The TBSM 4.2 Dashboard and Data servers might have to be restarted to pick up the policy changes for all services.

Creating the custom scorecard page

Now that the various pieces are created, the last step is to create a page for the executive user that becomes that executive's Executive Dashboard. A user with roles "iscadmins, tbsmAdminUser, and ncw_admin" are required to create this page.
The following steps are required to complete the custom Web page for executive user **ibank**:

1. Log in to the TBSM 4.2 console with a user ID that has the roles "iscadmins", "tbsmAdminUser", and "ncw_admin".
2. From the task list on the left side of the TBSM 4.2 Console, expand the **Settings** folder and select **Page Management**.

3. Select the **New Page** button.
4. From the **Choose a Portlet** window, select the radio button next to the portlet named Service Tree. Note that the user creating this page, in addition to having the role of "iscadmins", must also have the "tbsmAdminUser" role in order to see the TBSM portlets in the portlet selection list.
5. Select the **OK** button to exit the **Choose a Portlet** window. The new page with the default service tree is displayed.
6. In the row of buttons on the toolbar in the upper right corner of the page (beneath **Save** and **Cancel**), click the **Horizontal split** button ( ). This splits the page horizontally so that another portlet can be added.
7. From the **Choose a Portlet** window, select the radio button next to the portlet named Inline Frame. Note that the user creating this page, in addition to having the role "iscadmins", must also have the "ncw_admin" role in order to see the Inline Frame portlet in the portlet selection list.
8. Select the **OK** button to exit the **Choose a Portlet** window. The new page with the default service tree at the top and the default URL at the bottom is displayed.
9. Select the **Save** button in the upper right corner of the page.
10. In the Page name field, enter Internet Banking. Note the default location of the page in the Page location field or use the **Location** button to open a window that lets you move the page to the top of the navigation tree or under any existing folder.
11. Select **Roles with Access to This Page**, then select the **Add** button. From the list of roles, select "tbsmReadOnlyUser", which will match the role given to the user **ibank**. Note that a unique role could be defined and assigned to user **ibank** and to this page if other TBSM users should not be allowed to view the page.
12. To complete the creation of the **Internet Banking** page, click **Save**.

    At this point the page should be displayed in the TBSM 4.2 console, as shown in Figure 24. These final steps complete the customization of the page:

1. Locate the edit icons for each portlet in the page. The red boxes in Figure 24 illustrate where the edit icons are located.
2. Click the edit icon, then click **Edit Defaults** for the Service Tree portlet. To see the customized service tree, select **InternetBankingTree** from the Tree Template drop-down list.
3. Click **Save** when the updates are complete.
4. Now log out and log in with executive user **ibank**. The task list on the left side of the TBSM 4.2 Console should include a navigation element for "Internet Banking". You might have to expand one or more folders depending on the location you defined for the page.
5. Click the edit icon for the Inline Frame portlet. Specify a value for the URL field. In this example, a popular financial Web page was selected.

    The customized Web page shown in Figure 13 should now be displayed.

Troubleshooting

Table 5 lists common problems that might be encountered while configuring and using the TBSM 3.1 ESDA feature. Possible solutions are listed in the second column, and are references to the information found in Table 6.

Table 5: Common problems

Problem	Solution identifiers
Templates, services, data source, or data fetcher are not displayed in the TBSM 4.2 console after configuration.	Tree, Config
When the top level service is expanded, child services are not displayed. Either no child services are displayed or the wrong child services are displayed.	ChildBS, Data, TBSM31, ID
Business system folders and shortcuts added to or removed from a parent business system folder are not displaying in TBSM 4.2.	Inv, Data, TBSM31
The status for the TBSM 4.2 service does not match the corresponding TBSM 3.1 business system folder, or the status is updated too slowly.	Data, TBSM31, Fetch, Refresh
The right-click menus are not displaying properly or are not functioning correctly. The "Launch TBSM 3.1" action is missing or is not launching the TBSM 3.1 Web Console.	ActCfg, CSCfg

Table 6 contains possible solutions for the common problems. The first column contains an identifier for the solution and the second column has the details.

Table 6: Solutions

Solution identifier	Solution details
Tree	If services are not displayed, refresh the Service Navigation portlet on the Service Administration page to update the service tree.
Config	Run the configuration scripts again and note any error messages. Check that no syntax problems were introduced (for example, missing quotation marks) when the script was customized.
ChildBS	Make sure that the top level business system folder in TBSM 3.1 has business system folders as dependents. Business system resources are *not* displayed in the 4.2 service model.
Data	Make sure the **TBSM** data source can be connected. Use the Test Connection button on the "Edit TBSM" page and make any corrections required to the Username, Password, Host Name, and Port.
TBSM31	If the **TBSM** data source is defined correctly, make sure the TBSM 3.1 database server is available. If the TBSM 3.1 database server has been restarted, it might be necessary to restart the TBSM 4.2 Data server to restore the connection.
ID	Verify that the correct TBSM 3.1 id has been specified for the seed service that was defined in the tbsm31esda_selected_seeds.radsh file.
Inv	Make sure the invalidation period has expired for the 4.2 parent service that has had the change in dependencies in TBSM 3.1. The default invalidation period is 60 seconds, but this value might have been changed while configuring. Use the service editor page to manually invalidate the parent service, refresh the service tree, then expand the parent service again using the +.
Fetch	Check the data fetcher log using the **Show Log** function that is displayed when you right-click **TBSM_AlertStatus** on the Data Fetcher navigation view in the Service Navigation portlet. This

	portlet is found on the Service Administration page. If there are errors, check the data source connection. This can be a timing issue if the fetcher runs just before a service is added by the ESDA rule. If the polling interval has been increased, this also affects the timing of the status update. The fetcher can be run immediately by clicking the **Fetch Now** item on the right-click menu for **TBSM_AlertStatus**on the Data Fetcher navigation view. This view is available from the Service Navigation portlet of the Service Administration page.
Refresh	In addition to the polling interval for a data fetcher, there are intervals defined for refreshes of the service tree and all canvas views. This might be causing the delay in updating the TBSM 4.2 service status with the TBSM 3.1 status.
ActCfg	The configuration to add the "Launch TBSM 3.1" action to the right-click menu was not completed correctly. Restore the original files containing the action and view definitions and repeat the configuration steps found in 2.3.1, "Launching the TBSM 3.1 Web Console".
CSCfg	The additional attribute **consoleServer** must be set to the host name or IP address of the TBSM 3.1 console server. This value, or the **consoleServerPort** value, might not have been set properly when configuring the templates for this feature. Use the **Additional** tab to update this attribute's value on the Template editor page for the **TBSM31_Model** template.

Summary

The TBSM 3.1 ESDA feature provides rules that retrieve high level TBSM 3.1 business system folders and create service instances in TBSM 4.2. This helps preserve the customer investment in building the TBSM 3.1 business systems tree by reflecting at least the executive level business systems in a TBSM 4.2 service model. In addition, the TBSM 3.1 status can be reflected on the 4.2 services. Thus, TBSM 3.1 and TBSM 4.2 can be run in parallel and will stay consistent with regards to the high level service content and status.

Beyond simply reproducing the TBSM 3.1 high level service model, this feature exposes the model to all the many capabilities that TBSM 4.2 has to offer. By exploiting the SLA, data integration, and visualization capabilities of 4.2, TBSM 3.1 executive users can be provided with a greatly enriched Executive Dashboard.

Appendix

Sample SQL Data for Internet Banking Example

The following SQL code can be used to create the sample KPIData table and insert several rows of sample KPI data in the PostgreSQL database that is installed by TBSM4.2. This SQL code should be applied against a database named KPI to match the example in this paper.

```
--
-- Create Table
--
CREATE TABLE KPIData (
       servicename       VARCHAR(50) NOT NULL,
       kpitype           VARCHAR(50) NOT NULL,
       kpivalue          INTEGER NOT NULL
);

--
-- Insert sample data
--
```

```
INSERT INTO KPIData values('Northeast', 'Active', 1500);
INSERT INTO KPIData values('Northwest', 'Active', 700);
INSERT INTO KPIData values('Southeast', 'Active', 1000);
INSERT INTO KPIData values('Southwest', 'Active', 1200);

INSERT INTO KPIData values('Northeast', 'New', 25);
INSERT INTO KPIData values('Northwest', 'New', 15);
INSERT INTO KPIData values('Southeast', 'New', 35);
INSERT INTO KPIData values('Southwest', 'New', 5);

INSERT INTO KPIData values('Northeast', 'Closed', 45);
INSERT INTO KPIData values('Northwest', 'Closed', 10);
INSERT INTO KPIData values('Southeast', 'Closed', 5);
INSERT INTO KPIData values('Southwest', 'Closed', 8);

INSERT INTO KPIData values('Northeast', 'Transactions', 450000);
INSERT INTO KPIData values('Northwest', 'Transactions', 1000234);
INSERT INTO KPIData values('Southeast', 'Transactions', 50000);
INSERT INTO KPIData values('Southwest', 'Transactions', 183456);

INSERT INTO KPIData values('Northeast', 'Down', 50);
INSERT INTO KPIData values('Northwest', 'Down', 15);
INSERT INTO KPIData values('Southeast', 'Down', 100);
INSERT INTO KPIData values('Southwest', 'Down', 600);
```

Numerical Formula Rule

Table 3, "Template rules defined for child services of InternetBanking", defined the rule **Health** as a Numerical Formula rule that uses a policy. Following is the policy that was created for the example used in this paper. Look for comments containing "TBSM 3.1 ESDA Feature".

Tree Template Display

In the section entitled "Tree Template policy" on page 5, customization was described that formats the service tree display. Following is the content of file**TBSM_DATA_SERVER_HOME/policy/RAD_GetTreeColumnValue.ipl**, which is the policy that was modified in order to format the custom service tree. Look for comments containing "TBSM 3.1 ESDA Feature". This policy should be edited using the **Edit Policy** button found on the Tree Template editor.

References

- IBM Tivoli Business Systems Manager Version 3.1, *Introducing the Consoles*, __SC32-9086-00
- IBM Tivoli Business Service Manager Version 4.2, *Service Configuration Guide,* SC23-6041-02
- IBM Tivoli Business Service Manager Version 4.2, Customization Guide, SC23-6042-02
- IBM Tivoli Business Service Manager Version 4.2, Administrator's Guide , SC23-6040-02

For TBSM 4.2 publications, use a web browser to access:
http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/welcome.htm

For TBSM 3.1 publications, use a web browser to access:
http://publib.boulder.ibm.com/tividd/td/BusinessSystemsManager3.1.html

Attachments

The following attachments are available for download with this paper:

- canvasDynamicSubMenuActions.xml.update
- tbsm31esda_config.radsh
- ViewDefinition_.xml.update
- canvasOpenURLActions.xml.update
- tbsm31esda_config_shortcuts.radsh
- tbsm31esda_selected_seeds.radsh
- tbsm31esda_single_seed.radsh
- TBSM31_bussys_svg.gif
- TBSM31_bussys.svg

ITCAM for SOA and TBSM integration

his scenario describes how you can use TBSM to discover and monitor service objects based on data from IBM Tivoli Composite Application Manager for Service Oriented Architecture (ITCAM for SOA). In this scenario, you set up TBSM to work with data from the ITCAM for SOA using the Discovery Library toolkit and the Tivoli Event Integration Facility Probe (EIF Probe) for IBM Tivoli Netcool.

In this scenario, TBSM monitors and discovers services as follows:

- TBSM collects the application alerts from the Netcool/OMNIbus EIF Probe connected to the TEMS for the ITCAM for SOA host. You need to setup the TEMS to forward situation events to the EIF Probe and customize the EIF Probe to handle the SOA events.
- The TBSM Discovery Library toolkit collects the application data from the IBM Tivoli Enterprise Monitoring Server (TEMS) and IBM Tivoli Enterprise Portal Server (TEPS) connected to ITCAM for SOA. You need to set up the ITCAM for SOA data feed and customize the Discover Library toolkit to handle the SOA data.

How TBSM identifies services

When the EIF Probe filters situation events, it converts them to Netcool/OMNIBus ObjectServer events that can be read by TBSM. TBSM identifies the service-affecting events with the value of the **BSM_Identity** field. You edit the EIF Probe rules file to automatically create a unique **BSM_Identity** value for each service you want to monitor.

Conversely, when the Discovery Library Toolkit discovers a service from TADDM or a DLA, it creates a unique **BSM_Identify** field value for each discovered service. You edit the eventidentifiers.xml file to automatically create **BSM_identify** field values that match the values created by the EIF Probe.

When the **BSM_Identity** field/value pairs match, TBSM discovers and monitors the services associated with your ITCAM for SOA environment.

Prerequisites

Before you begin, you need to install the ITCAM for SOA application and associated components.

- Install IBM Tivoli Monitoring 6.1 FP4 (or higher) with Tivoli Enterprise Portal Server extensions.

- Install ITCAM for SOA v6.1 or higher and the Discovery Library Adapter (DLA). See the ITCAM for SOA information center.

- Install the Tivoli EIF Probe. See the TBSM information center.

Step 1: Forward TEMS events to the EIF probe

To monitor the status of TBSM services created from the ITCAM for SOA data, you need to configure the IBM Tivoli Monitoring to forward events to an IBM Tivoli Netcool/OMNIbus ObjectServer. The TEMS connects to the EIF Probe as another Tivoli Enterprise Console server.

Events flow from the hub Tivoli Enterprise Monitoring Server (TEMS) to the Tivoli EIF Probe. The EIF Probe filters the events and forwards the events to the Netcool/OMNIbus ObjectServer monitored by TBSM. For this scenario, you can use the events for your ITCAM for SOA monitoring services in your production or test environments.

To configure the IBM Tivoli Monitoring to forward ITCAM for SOA events to TBSM, complete the following steps:

1. Open the Manage Tivoli Enterprise Monitoring Services application.
2. Right-click on the service **Tivoli Enterprise Monitoring Server** and select **Reconfigure** from the menu.
3. On the TEMS configuration options window, enable the **TEC Event Integration Facility** check box.
4. Click **OK** twice. The TEC Server: Location and Port Number window opens.
5. In the TEC Server Location field, enter the host name where the EIF Probe was installed.
6. In the TEC port number field, enter the port number of the EIF Probe.
7. Click **OK** to close the window.

Step 2: Edit the TEMS EIF configuration files

1. On the IBM Tivoli Monitoring host, change to the directory where the TEMS is installed. The default directory is: **Windows**: \IBM\TM\cms\TECLIB\

   UNIX: <itm_installdir>/tables/host name/TECLIB/om_tec.config
2. Make a backp copy of the original OM_TEC.CONFIG file.
3. In a text editor open the file: OM_TEC.CONFIG.
4. Add the properties and values described in the following table:

Property Name	**Value**	**Description**
FilterMode	IN	Enables event filtering.
Filter:Class	ITM_Services_Inventory_610;	Event filters that forward all ITCAM for SOA events for the Services_Inventory_610 table to the Tivoli EIF Probe.
FilterCache:Class	ITM_Services_Inventory_610;	Filter that specifies which events to cache when IBM Tivoli Monitoring is unable to forward an event to the Tivoli EIF Probe

5. Save and close the file.
6. Restart the Tivoli Enterprise Monitoring Server

Note: If the EIF configuration file already contains values for any of the parameters listed above, you should replace the current values with the values described in the table.

Step 3: Map situation events

The EIF probe maps the ITCAM for SOA situation events to Netcool/OMNIbus event tokens. The BSM_Identity strings created from these event tokens must match the BSM_Identity strings that were created from the**EventIdentifiers.xml** file for the Discovery Library toolkit.

By default, the situation events the EIF probe receives do not map all the required event tokens needed to identify the ITCAM for SOA services. The key event tokens excluded are:

- operation_name_u
- operation_namespace_u
- service_name_u
- port_namespace_u
- service_type

These event tokens uniquely identify a specific Service Port/Operation as defined by ITCAM for SOA. To map these event tokens, modify the tivoli_eif.rules file to build a BSM_Identity event token that maps to a specific TBSM service instance (in this case, an SOA resource). You make the changes in two different places in the tivoli_eif.rules file.
On Windows systems, the rules files is in the **%NCHOME%\omnibus\probes\win32** directory.
On UNIX systems, the path is **$NCHOME/probes/<os_name>**.

Before you begin, you need to Forward TEMS events to the EIF probe.

Customizing event token mapping

In this example, you create a logical block to remove single quotes from custom SOA event tokens. In this block all the events tokens used by the probe are stripped of single quotes which is required to avoid problems in later processing.

You also create an SOA-specific BSM_Identity token.

To configure the EIF Probes rule file with custom event tokens:

1. Open the **tivoli_eif.rules** file in a text editor.
2. Find the line with the text: **#End removal of embedded single quotes**.
3. To remove the single quotes from the custom SOA event tokens, enter the following code before the **#End removal of embedded single quotes** comments.

```
if(exists($operation_name_u ))
 {
                        if(regmatch($operation_name_u , "^'.*'$"))
                        {
                                $operation_name_u = extract($operation_name_u ,
"^'(.*)'$")
                        }
 }
 if(exists($operation_namespace_u ))
 {
                        if(regmatch($operation_namespace_u , "^'.*'$"))
```

```
                              {
 $operation_namespace_u = extract($operation_namespace_u , "^'(.*)'$")
                              }
 }
 if(exists($port_namespace_u ))
 {
                              if(regmatch($port_namespace_u , "^'.*'$"))
                              {
                                        $port_namespace_u = extract($port_namespace_u ,
"^'(.*)'$")
                              }
 }

 '''if(exists($service_name_u ))'''
 {
                              '''if(regmatch($service_name_u , "^'.*'$"))'''
                              {
                                        '''$service_name_u = extract($service_name_u ,
"^'(.*)'$")'''
                              }
 }
 if(exists($service_type ))
 {
                              if(regmatch($service_type , "^'.*'$"))
                              {
                                        $service_type = extract($service_type , "^'(.*)'$")
                              }
 }
```

The quotes are stripped from the new SOA event tokens the same way they are for the default event tokens. The code identifies the specific event token (such as **$service_name_u**), and applies functions (regmatch and extract) and regular expression to the token to remove the single quotes.

4. To create the BSM_Identify field value from the custom SOA event tokens, enter the following code between the last two curly brackets at the end of the rules file:
   In this sample, the BSM_Identity field is build from the values for the operation_name_u, operation_namespace_u, service_name_u, and port_namespace_u SOA event tokens. **Note:** ITCAM for SOA sends a situation event for both the requestor and provider of the service port/operation pair. For this integration purpose you only filter events/situations the provider side by adding a check in the rules file for service_type=1.

5. Save and close the file.

6. To enable the changes, stop and restart the EIF Probe. In Windows, start the Tivoli_EIF probe service. In UNIX, execute the nco_p_tivoli_eif script from the **$NCHOME/omnibus/probes** directory.

Step 4: Customizing the Discovery Library toolkit

The Discovery Library toolkit filters ITCAM for SOA data and TBSM creates service instances based on the SOA data. The data can come from a DLA book file or a feed from TADDM host. To correctly manage the ITCAM for SOA data, you need to modify these XML control files on the TBSM host. These files are located in the following directory:

$NCHOME/XMLToolkit/xml

Event Identifier rules file

The **EventIdentifierRules.xml** file in the Discovery Library toolkit creates TBSM Identification field/value pairs for services imported from TADDM or DLA books. TBSM uses these identification values to correlate events to service instances. The default event identification rules provided with the Discovery Library toolkit map the IP Address or IBM Tivoli Managed managed system name to BSM_Identity values. To use the sample the integration with ITCAM for SOA, you need to customize these files.

Before you customize the event identification rule, backup the existing **EventIdentifierRules.xml** file. To handle the SOA data, you create new policy in the **EventIdentifierRules.xml** file. This Netcool/Impact policy defines an event identifier rule to match the status events from the SOA resource that were customized in the EIF Probe rules file.

1. In a text editor, open the **EventIdentifierRules.xml** file.
2. Search for the line: **<Mapping policy="GetAllIPAddressesAndFQDN" class="cdm:sys.OperatingSystem"/>**
3. Above this line, add the following code for the SOA event identifier policy:
   In the above policy for class type **WSOperation**, the event identification is built from a combination of the web services operation name, operation name space, port name and port name space.
4. Save and close the file.

   The above policy creates a TBSM BSM_Identification field value such as:

   As a result, each TBSM service instance created from the SOA data has a unique BSM_Identification field value. Whenever TBSM detects an event with this BSM_Identification field value, it will check the event for status information for the associated service.

   In this example, the event identifier policy creates BSM identifier with the operation name and port name.

   Using the above policy the BSM_Identification value is:

Labeling rules file

By default, TBSM creates the display name of the service instance from cdm:Label CDM attribute of the object. You can customize the **LabelingRules.xml** file to create the display name from other CDM attributes of the SOA object.

When you integrate TBSM with ITCAM for SOA, the service instance that was created is the name of the web service operation on a particular web service port. In TBSM, this value is the service display name. To maintain the uniqueness, you customize the name label to be:

```
web service operation name @ web service port
```

You also include the name and namespace attributes of both the objects.

Backup the existing **LabelingRules.xml** file and search for: <Mapping policy="Global" class="%" />

Add the policy in this code sample:

By performing the above customization, displayname of the service instance is generated as:

Composite rules file

The **CDM_TO_TBSM4x_MAP.xml** file groups a set of related objects into a composite representation of a single object. In some cases, the level of detail in the CCMDB is too granular. The composites create groups of objects to the level of management appropriate to the application. You can modify the default behavior of the CDM relationship types to a TBSM dependency to avoid the cyclical relationship between the various web services operations, web services ports and web services port end points.

Note: You need to have an in-depth understanding of the elements in the IBM Common Data Model before you attempt to change the composites configuration.

Composites are needed to group a set of related objects into a workable representation of a single object. If the level of detail in the CCMDB is too granular for certain applications, composites are used to create groups of objects at the level of management appropriate to the application. The default behavior of the CDM relationship types to a TBSM dependency was modified to avoid the cyclical relationship between the various web services operations, web services ports, and web services port end points. This is achieved by editing **CDM_TO_TBSM4x_MAP.xml** file in **%NCHOME%\XMLtoolkit\xml** directory.

Before you customize the file, backup the existing **CDM_TO_TBSM4x_MAP.xml** file. Search for following the string:

Add the lines from the following sample:

This change limits the relationship of cdm:invokedThrough has no dependency on target cdm.soa.WSPort on source cdm:soa.WSOperation. Similarly it also limits the dependency of cdm:soa.WSPort on cdm:soa.WSEndpoint.

Step 5: Processing the ITCAM for SOA DLA

The ITCAM for SOA 6.1 product shipped an ITCAM for SOA DLA that discovers SOA resources from your monitored environment. Follow the readme to discover services in your test or production environment. Run the ITCAM for SOA DLA following the documentation in the ITCAM for SOA DLA readme to generate an xml book with SOA resources in your monitored environment. A sample book (**ITCAMSOA61.cvtwin02.2007-05-01T02.13.52Z.refresh.xml**) discovered on our test environment is included in the TBSM_ITCAM4SOA.zip file shipped with the ITCAM for SOA Status Event Integration white paper at the TBSM Open Process Automation Library (OPAL).

There are two ways to process the IDML book:

- TBSM 4.1 Discovery Library Toolkit directly processing the IDML book generated by the SOA DLA
- Load the book into TADDM and TBSM 4.1 Discovery Library Toolkit integrates with TADDM and gets the information from TADDM

Process the DLA book

Copy the IDML book generated in the section above to **%NCHOME%\dlbooks** directory on TBSM Server which is the discovery library file system. Ensure the discovery library toolkit service is started. The discovery library toolkit service consumes the idml book from ITCAM for SOA and populates the service component registry with SOA resources.

The processing of the IDML book can be validated by checking the **msgGTM_XT.log** file for the message similar to the following:

```
GTMCL5290I: Book ITCAMSOA61.cvtwin02.2007-03-09T23.24.54Z.refresh.xml processed
successfully.
```

The following path indicates the default location of the log file:

- **Windows:** %NCHOME%\XMLtoolkit\log
- **UNIX:** $NCHOME/XMLtoolkit/log

Step 6: Create the TBSM service instance

Once the book is processed and the service component registry (SCR) is populated, the objects created in the SCR can be used to construct a business service in TBSM depending upon the requirement of service model.

Complete the following steps to create a service instance:

1. Login to TBSM console as a user with service administration permissions.
2. Select **Service Administration** from the Pages list and click **Go**.
3. Click **Templates** tab and click **Create New Template**.The service template describes a class of services. When you create the new service instance, you assign to this new template.
4. Give the name of the template as **ACMEServicesTemplate**.
5. In the Rules tab, click **Create a dependency rule**. This launches a dependency rule editor.
6. Name the dependency rule as **BSM_SOAOperationPortsWorst** with a Child Rule/Mapping as BSM_SOAOperationPort and click **OK**.
7. Save the new template.
8. Next, create a business service and assign it to the new service template. To add the services from SCR, click the **Dependents** tab and are click the **Add from Service Component Registry** button.
9. Select the SCR services created from the SOA DLA book.
10. When you have finished adding services, save the service template.

Step 7: Firing situation events

ITCAM for SOA has predefined situations that can be used to monitor service performance metrics. Use the Response Time Critical situations to illustrate the integration between TBSM and ITCAM for SOA. The Response Time Critical situation has a pre-defined value of 10 seconds. This means that when response to a web service exceeds 10 seconds a Response Time Critical situation will be fired by the IBM Tivoli monitoring infrastructure.

To trigger the situation, invoke the echoWithDelay operation using the web services sample application user interface with any string and a delay of greater than 10 seconds.

By default the sampling interval is five minutes, so one request with a response of 20 seconds in a five minute period should be enough to fire the event providing you make no other shorter requests during the same interval. This situation will show up as shown below in the Tivoli Enterprise Portal console.

When the response time critical situation fires in ITCAM for SOA an EIF event is sent to the Tivoli EIF probe which forwards that event to the OMNIbus server. This event is processed by the server and the SOA resource's status changes to RED because it is a critical situation that fired.

The following is the screen shot on the TBSM Service Tree:

Once the situation from ITCAM for SOA is cleared, ITM sends a clearing situation to TBSM and the ACME Business Service that is shown as red turns back to green. There are some additional customizations that can be done for the look and feel aspect of this integration to customize the flyover text of service instances in TBSM.

Launch in Context

If a Tivoli Business Service Manager (TBSM) service was created from a Discovery Library Adapter (DLA) book generated by IBM Tivoli Monitoring, you can launch into the Tivoli Enterprise Portal from the TBSM service. This function is only supported in Internet Explorer on the Windows operating system.

The **Launch in Context to TEP** menu option is enabled only if the service contains additional attributes that contain the data used to construct a URL for the Enterprise Portal web console. There are two attributes created as part of the TBSM Discovery Library Toolkit service creation process. The Discovery Library Toolkit obtains the launch data from either an IDML book produced by the Tivoli Monitoring Services DLA or from the Tivoli Application Dependency Discovery Manager. The user can create their own services and provide valid and correct values to these attributes.

The **IBM_Tivoli_Monitoring_Services_sourceContactInfo** attribute contains the host name and the port number of the Tivoli Enterprise Portal associated with the TBSM service.

For example:

The **IBM_Tivoli_Monitoring_Services_sourceToken** attribute contains the information about the resource in the Tivoli Enterprise Portal.

For example:

The values for these attributes come from the DLA book for Tivoli Monitoring or from the Tivoli Application Dependency Discovery Manager. Any service which is not an associated Tivoli Monitoring resource does not have the above attributes.

Installing launch in context for a client

To enable the launch in context feature on a TBSM client or a client with the Active Event List (AEL) for IBM Tivoli Netcool/Webtop, you need to install and configure some files as described in the installation section of the TBSM information center.

Launching Tivoli Enterprise Portal from a service

To enable the launch in context feature on a TBSM client or a client with the Active Event List (AEL) for IBM Tivoli Netcool/Webtop, you need to install and configure some files as described in the installation section of the TBSM information center.

To launch the resource in Tivoli Enterprise Portal from TBSM, complete the following steps:

1. In the Service tree or Service Viewer, right-click the service.
2. From the menu that opens, right-click Integrations.
3. From the menu that opens, click **Launch TEP**.
   When the launch in context is issued for the first time, a new browser opens, providing a link for asisp_launch_browser.exe.
4. Download the .exe file and copy it to C:\Program Files\Internet Explorer directory as directed.
5. Issue the **Launch TEP** option again. A new browser opens.
6. Enter the Tivoli Enterprise Portal user ID and password.
   After you enter your user ID and password, the resource is displayed in the Tivoli Enterprise Portal.

Launching Event Displays with Dashboard Widgets

This paper and accompanying video are a guide to configuring IBM® Tivoli® Business Service Manager (TBSM) to launch event viewers unique to a widget on a custom canvas dashboard.

Normally, a *widget*, an indicator that graphically represents the status of a business service on a TBSM dashboard, is the collection point of events that affect its status. Finding the events that resulted in its current status is as easy as a right-click to display service effecting events on an Active Event List (AEL) display.

However, if you are structuring your dashboard differently, for example, using a custom canvas for your dashboard and widgets that derive their status in more complex ways by querying for a particular set of events, displaying those events can be more difficult to do. This paper explores modifying the single-click behavior of dashboard widgets on a custom canvas dashboard to automatically display an AEL page to view the events that make up the status of the widget.

To read this paper, click here.

To view the accompanying video, click here.

About the Authors

This paper was written by David Schmidt, an integration architect for Tivoli Service Availability and Performance Management Solutions, and Phil Riedel, an integration architect for Tivoli Service Availability and Performance Management Integration Center of Competency.

Monitoring Web Applications through Transactions

Quick Configuration Guide for SSO

Quick Configuration Guide for SSO

The purpose of this paper is to detail the post installation configuration of IBM® Tivoli® Netcool® OMNIbus WebGUI, IBM Tivoli Business Service Manager (TBSM), and IBM Tivoli Monitoring with LDAP and implement Single-Sign-On (SSO) using Lightweight Third-Party Authentication (LTPA) between these three products.

The environment described in this paper uses the following product versions:

- IBM Tivoli Netcool OMNIbus WebGUI version 7.3
- IBM Tivoli Business Service Manager version 4.2.1 Fix pack 2
- IBM Tivoli Monitoring version 6.2.2 Fix pack 2

All products were installed on an AIX® system using the file based repository authentication option.

To read this paper by Daniel Rhames, click here.

Solution Guide for Integrating Tivoli System Automation Application Manager with Tivoli Business Service Manager

Typically, business applications consist of different middleware components, are multi-tiered, and runon heterogeneous platforms. IBM® Tivoli® Business Service Manager (TBSM) is used to providehealth information of the multi-tiered application and to provide monitoring of service level agreements(SLA) based on information coming from numerous sources.

IBM Tivoli Netcool®/OMNIbus is used to collect all events that are related to the business applicationlandscape, and TBSM uses these events to determine the status of the business applications.

IBM Tivoli System Automation Application Manager is used in business application landscapes toautomate start and stop dependencies, provide a common operating environment, provide automaticrecovery in failure situations, and to get an aggregated availability status.

IBM Tivoli System Automation for Multiplatforms and IBM Tivoli System Automation for z/OS areused to make individual components of the business application highly available, for instance, a criticaldatabase.

This document presents a solution showing the integration of IBM Tivoli Business Service Manager(TBSM) and Tivoli System Automation Application Manager to help you monitor, control, andautomate your business applications from an end-to-end point of view.

The document is divided into the following parts:

Part 1: Integration overview and usage scenarios
Describes the usage scenarios of the integration solution.

Part 2: Step-by-step setup instructions
Describes the required customizations in TBSM and Tivoli System Automation Application Manager tocreate the integration solution.

Click here to read the entire document.

TBSM Discovery Library Toolkit Configuration - Integrating with TADDM 7.3

TADDM 7.3 has been released. In order to connect TBSM to a TADDM 7.3 server, you must install TBSM 6.1.1 Fixpack 2. This should be done prior to allowing the TBSM Discovery Library Toolkit to connect to the TADDM 7.3 server.

For import of Business Applications from TADDM 7.3 to TBSM 6.1.1, the property com.ibm.cdb.serviceinfrastructure.earlier.ver.compatibility must be set to true in collation.properties. This will be true by default for upgraded TADDM 7.3 servers. It is false, by default, for fresh TADDM 7.3 installs. Note: TADDM restart is required after property modification.

If com.ibm.cdb.serviceinfrastructure.earlier.ver.compatibility is flipped from false to true, then all Business Applications need to be re-build, to regenerate backward compatible Business Applications. This can be done by either waiting for scheduler to process all Grouping Patterns (and recreate new and old Business Applications) or by manually executing (all or selected) Grouping Patterns from TADDM UI (or command line).

In addition, depending on how you have configured the TBSM/TADDM integration, you may need to migrate the configuration. For more information on migration, see
Migrating the TBSM Discovery Library Toolkit for Integration with TADDM 7.3.

TBSM Discovery Library Toolkit Configuration-How to add new BSM Identity Values

This document shows how to edit Discovery Library Toolkit configuration xml, to add a new BSM Identity. The documentation links below are for TBSM 611 documentation. If you are using 610, please use the documentation for that version here.

Before you start, you will need the following:

1. Name of cdm class for which the new Identifier is to be added.

2. Name of attribute(s), if any, to be used for creating the identifier.

3. (Optional) If attributes from a related instance are to be used, then we need what?

The easiest way to gather this information is to use CRViewer to look at the data loaded into the SCR by the Discovery Library Toolkit, from either TADDM or IDML books. Documentation on starting the CRViewer is here.

Within CRViewer, select the appropriate class on the top left section. Next select any instance of that class in the lower left section. Selecting an instance automatically populates the attribute list (top right section). Click on **Get Details** to see more details for the attributes (lower right section). Use the Identifiers tab to see already configured Identifiers.

Next retrieve the current configuration file (EventIdentifierRules.xml) from the TBSM database using getArtifact. See documentation here.

Edit the retrieved file, EventIdentifierRules.xml, by adding a new policy section and a new mapping for the policy, for example

```
<Policy name="ANewPolicyCallMeWhatYouLike">
   <Rule name="ANewRuleWithAMeaningfulName">
      <Token keyword="LITERAL" value="XXX"/>
   </Rule>
</Policy>
```

<Mapping policy="ANewPolicyCallMeWhatYouLike" class="CDM Class Name" />

The above will add a BSM Identify value of "XXX" for all instances of class "CDM Class Name"

If you prefer to use an attribute from the attribute list in CRViewer, for example cdm:Label, then use Token keyword ATTR, as follows:

```
<Policy name="PolicyForCDMLabel">
   <Rule name="RuleForCDMLabel">
      <Token keyword="ATTR" value="cdm:Label"/>
   </Rule>
</Policy>
```

<Mapping policy="PolicyForCDMLabel" class="CDM Class Name" />

Many tokens can be concatenated together, for example:

```
<Policy name="PolicyForUsingTwoTokens">
   <Rule name="RuleforUsingTwoTokens">
      <Token keyword="LITERAL" value="XXX"/>
      <Token keyword="ATTR" value="cdm:Label"/>
   </Rule>
</Policy>
```

<Mapping policy="PolicyForUsingTwoTokens" class="CDM Class Name" />

When the change(s) to EventIdentifiers.xml is complete, save the artifact to the TBSM database using dbfileutility and reload the CDM definitions. See here for how to use utils to reload CDM definistions. A restart of the Discovery Library Toolkit may be required.

Further documentation on customizing Discovery Library Toolkit artifacts is available here.

Use related objects to create BSM Identity values

The above examples show how to use class attributes to create BSM Identity values. However, it is also possible to use attributes from a related class instance to generate Identifiers. If you have a relationship between 2 objects you can use that relationship to jump to the other object to obtain an attribute.

As before, you need to add a Policy and a Policy Mapping. For the Policy Mapping, map the class you want the BSM Identity to be associated with to the policy.

For the policy, add a Relationship tag for the relationship between this class and the class which actually contains the attribute. For example, if you want to add a new BSM Identity for an object of cdm class "Orange", which uses an attribute of a related object, of cdm class "Purple", then the policy follows the orange object through the relationship type it has and the direction of that relationship to the purple class. Once there, output the attribute is obtained. Here is an example of how to obtain the cdm:Name attribute from an object of class "purpleclass" which is related to "orangeclass" through the relationship 'cdm:memberOf'. the source of the relationship is "purpleclass". CRViewer shows relationships in the Relationship tab in the lower left section.

```
<Mapping policy='policyorange' class = 'orangeclass'/>

<Policy name='policyorange'>
 <Rule name='ruleorange'>
  <Relationship relationship='cdm:memberOf' relationshipSource='purpleclass'>
     <Token keyword='ATTR' value='cdm:Name'/>
    </Relationship>
 </Rule>
</Policy>
```

Tivoli Application Dependency Discovery Manager and Tivoli Business Service Manager Integration Solution Guide

IBM® Tivoli® Application Dependency Discovery Manager (TADDM) and Tivoli Business Service Manager (TBSM) are designed to work together, and their primary functions compliment each other. Configuring this integration requires some steps be taken during installation and some post configuration steps that are presented in this document.

The core value of the TADDM/TBSM integration is in complimenting our monitoring ability to identify the need for the provisioning-of/removal-of servers to a cluster and then leveraging the TADDM discovery process to automatically add/remove/change these resources in the TBSM service tree/viewer. - Integration Best Practices

This solution guide will demonstrate the Virtual Center support and show the ability to update TBSM portlets for Custom Servers and Business Applications that are discovered by TADDM.

TADDM Virtual Center Discovery and steps required for integration with TBSM

TADDM discoveries of Virtual Centers

If you want to discover Virtual Centers (VC) using TADDM, the following pages describe the additional steps that are required in TBSM for integration. You can also see how the VC systems are represented in both TADDM and TBSM.
Figure 1 shows the TADDM product console with the topology of a Virtual Center Server displayed. Figure 2 shows a similar view of the same object in TBSM.

To enable a Virtual Console discovery, do the following steps:

1. Add a custom profile and enable the following sensor.

2. You must also add an access list entry for the Virtual Console system.
   Navigate in the TADDM Product Console to **Discovery > Discovery Profiles**.

3. Add a new profile, model it after Level 3 discovery, and enable the sensor .

   Figure 4 is a TADDM Topology View of a Virtual Center (partial), showing virtual topology on the Virtual Center Server CI named x86ondemand14.tivlab.austin.ibm.com.

   The following TBSM changes are required for Virtual Center/Systems on TBSM 4.2.1:

1. After discovering new VC relationships or systems, run an. /explicitrel 0 command from the TADDM /dist/bin directory. All new relationship types in the CMDB are then available in TBSM.

2. Make the following changes in TBSM to support Virtual components:
   Add the following four lines that are highlighted in bold to the XMLtoolkit/xml/CDM_TO_TBSM4x_MAP_Templates.xml file. You do not have to add these lines at the end of the file as shown in the example.

   Add the following line near the end of the file XMLtoolkit/config/filters/classfilters.xml file, prior to </baseclass> entry.
   <baseclass import='true' class='com.collation.platform.model. topology.sys.vmware.DataCenter' depth='0'>

Defining Business Services: 2010 Best Practices with TBSM and TADDM

- The best practice is to use TBSM for defining business services. You can use the TBSM DLA to load these business services into TADDM.

- TADDM should be used for discovery and can be used for application creation and management as shown in this document.

TADDM and TBSM integration steps

Do the following procedures to integrate TADDM and TBSM:

1. Install TADDM and TBSM. This is not shown.

2. Configure the XML toolkit in TBSM. Enable TADDM for sending events. Some of these steps are done during installation.

3. Add a custom server template in TADDM and then run a discovery including these servers.

4. Add an application template in TADDM (for the custom server objects discovered previously) and run another discovery.

5. Define a new page/view in TBSM, with a service tree for the application that you added in step 4, an active event list, and a custom canvas.

6. Figures 14 and 15 show the removal of a server from your mail application using the TADDM dormant components delete function. Your service tree, event list and dashboard automatically reflect the removal of the server in TBSM.

7. Dynamic updates require that you set the XML toolkit to process updates at specified intervals.

8. Launch-in-context (LIC) from TBSM to TADDM is shown in Figure 23.

Creating a Custom Server in TADDM

1. Run an initial discovery.

2. Select the Mail Servers that are discovered to an application template in TADDM named Mail Applications.

Create your Business Service in TBSM and not in TADDM (this is a best practice). Figure 10 shows the TADDM creation of a business service, this step should not be performed.

Creating a Business Service in TBSM for the Mail Applications

TBSM by default attempts to load the "most commonly used classes" from the TADDM CMDB. You want to plan your TBSM implementation carefully:

- Plan for the business services you will need
- Plan for and include the appropriate CI's in your TADDM applications

Filter unneeded classes using the XMLtoolkit classfilters.xml file. This will improve the overall performance and user experience of this integration. Best practice is to be overly aggressive with filtering; allowing classes back in based
on business management objectives.

http://ibm.biz/IntegratingTBSMAndTADDM

Figure 12 shows the tree template editor (use for events and policies).

Configuring TBSM to pull data from TADDM

To see changes in TBSM you must pull data from TADDM. Use the XML toolkit by running the cmdbdiscovery.sh command (for this demonstration, this is done automatically by way of the polling interval):

1. Before running the toolkit, copy two files from TADDM. These files are in the /dist/sdk/lib directory. Copy them to /XMLtoolkit/sdk/clientlib/ directory on the TBSM data server. The two files are named platform-model.jar and taddm-api-client.jar.

2. The command and its properties file can be found in the /tivoli/tbsm/XMLtoolkit/bin directory. The properties file is named xmltoolkitsvc.properties. Issue the command on demand or use the cron command from the same directory.
The addition or removal of any system that is running the Custom Server automatically updates a custom dashboard template made from a subset of the Service Administration/Service Component Tree.

Important: TADDM discovery does not delete a system/application if it does not respond. However, TADDM provides the Dormant Components utility under Analytics. Therefore, if the Mail Servers become dormant, they can be deleted from this view.

Adding a TBSM created business service to the TADDM CMDB

Use the following procedure to add TBSM created business services to your configuration management data base.

1. On TBSM system change current directory to ../tbsm/XMLtoolkit/bin

2. Run the TBSM DLA command shown: genidml.sh -U bsmuser -P bsmupw -o /tbsm/XMLtoolkit/exportdla

3. Copy the output file to your TADDM server.

4. Start the TADDM non-root user. Change the directory to/dist/bin.

5. Run the ./loadidml.sh -f <TBSM DLA file> file name itbsm...refresh.xml command.

Delete using the TADDM Dormant Components function

Figure 15 shows before and after views of the custom tree. There are four systems in the first image. One will be removed in TADDM. The second view is displayed automatically after the TADDM discovery is complete. AEL shows events just for selected objects (borgy). The Business Application view in TADDM is also shown (after the delete).

Integrating Events between IBM Tivoli Monitoring and TBSM.

There are three steps in this process:

1. Run the Tivoli Monitoring DLA that is appropriate for your version of Tivoli Monitoring (tmsdla or KfwTmsDla). The DLA enables LIC to Tivoli Monitoring.
2. Load that DLA file into TADDM, TBSM, or both products.
3. Configure the Hub Tivoli Enterprise Monitoring Server (HTEMS) to forward events to the same probe that is used by TBSM/TADDM in this case.
4. Situation events from Tivoli Monitoring will automatically be associated with objects that are being monitored by Tivoli Monitoring. TADDM also contains a monitoring coverage report that lists the Tivoli Monitoring managed nodes.

Isolating your application for TBSM Dashboards formulas

Do the following steps to isolate your application:

1. From the TBSM rad/Postgres database using the following data fetcher SQL query: select * from view_components where label like 'Mail Servers'.
   This returns 7426.
2. Select * from view_dependencyrelations where srcid in (7426).
   This returns a list of HR systems or with count "*" the system count.

Creating a Service for the Dashboard (TestSub)

The associated Template (TestTemplate)

Adding New Pages in TBSM

Do the following steps to add new pages:

1. Click Page Management > New Page > Custom Page > Custom Tree. The new page icon is highlighted in Figure 20.
2. Add a custom name and select the check box if you want to show the starting node.
3. After adding, you can split the page from left to right or from upper and lower.
4. You can display your page to edit it. To link your tree segment to the AEL, click Action edit > show wires > new wires.

5. Be sure to enable the Source (Service Tree Object) Target (settings > page > AEL). Select the all events Entity.

Editing Dashboards: Porting/Inserting images

Do the following steps:

1. From the dashboard Portlet, click the Edit Preferences icon.

2. Click the Viewer tab and select the check boxes for all Visibility Controls.

3. Click Save.

1. Add any image, select it.

2. Click View > Inspector.

The image in Figure 22 was ported from TADDM (.,,/dist/images/custom/icon_customServer6_default.svg). Import the TADDM files(s) into TBSM (,../tip/systemApps/isclite.ear/sla.war/images/customServer6_default as a .png file type).

The icon location is then entered in the inspector for the selected dashboard image.

LIC from TBSM to TADDM (Change History)

Summary

Use TBSM to create and manage your business services and use TADDM to discover and automatically populate the components of those services with CI additions, deletions, and updates. TADDM provides an extensible discovery engine and the CMDB database. TBSM and its portlets provide a view of the business services, their health, and other key real time event/status information. TBSM and Tivoli Information Portal also provide you with the custom dashboard functions that are required to maintain an overall view of the health of your key business services.

For more information see http://ibm.biz/TADDMLink

About the Author

This Solution Guide was written by Phil Riedel, a Solution Architect in the Tivoli Service Availability and Performance Management Integration Center of Competency organization.

Notices

Tivoli Business Manager Time Window Analyzer and TADDM Change Event Integration Solution

Tivoli Composite Application Manager for SOA

This scenario describes how you can use TBSM to discover and monitor service objects based on data from IBM Tivoli Composite Application Manager for Service Oriented Architecture (ITCAM for SOA). In this scenario, you set up TBSM to work with data from the ITCAM for SOA using the Discovery Library toolkit and the Tivoli Event Integration Facility Probe (EIF Probe) for IBM Tivoli Netcool.

In this scenario, TBSM monitors and discovers services as follows:

- TBSM collects the application alerts from the Netcool/OMNIbus EIF Probe connected to the TEMS for the ITCAM for SOA host. You need to setup the TEMS to forward situation events to the EIF Probe and customize the EIF Probe to handle the SOA events.
- The TBSM Discovery Library toolkit collects the application data from the IBM Tivoli Enterprise Monitoring Server (TEMS) and IBM Tivoli Enterprise Portal Server (TEPS) connected to ITCAM for SOA. You need to set up the ITCAM for SOA data feed and customize the Discover Library toolkit to handle the SOA data.

How TBSM identifies services

When the EIF Probe filters situation events, it converts them to Netcool/OMNIBus ObjectServer events that can be read by TBSM. TBSM identifies the service-affecting events with the value of the **BSM_Identity** field. You edit the EIF Probe rules file to automatically create a unique **BSM_Identity** value for each service you want to monitor.

Conversely, when the Discovery Library Toolkit discovers a service from TADDM or a DLA, it creates a unique **BSM_Identify** field value for each discovered service. You edit the eventidentifiers.xml file to automatically create **BSM_identify** field values that match the values created by the EIF Probe.

When the **BSM_Identity** field/value pairs match, TBSM discovers and monitors the services associated with your ITCAM for SOA environment.

Prerequisites

Before you begin, you need to install the ITCAM for SOA application and associated components.

- Install IBM Tivoli Monitoring 6.1 FP4 (or higher) with Tivoli Enterprise Portal Server extensions.
- Install ITCAM for SOA v6.1 or higher and the Discovery Library Adapter (DLA). See the ITCAM for SOA information center.
- Install the Tivoli EIF Probe. See the TBSM information center.

Step 1: Forward TEMS events to the EIF probe

To monitor the status of TBSM services created from the ITCAM for SOA data, you need to configure the IBM Tivoli Monitoring to forward events to an IBM Tivoli Netcool/OMNIbus ObjectServer. The TEMS connects to the EIF Probe as another Tivoli Enterprise Console server.

Events flow from the hub Tivoli Enterprise Monitoring Server (TEMS) to the Tivoli EIF Probe. The EIF Probe filters the events and forwards the events to the Netcool/OMNIbus ObjectServer monitored by TBSM. For this scenario, you can use the events for your ITCAM for SOA monitoring services in your production or test environments.

To configure the IBM Tivoli Monitoring to forward ITCAM for SOA events to TBSM, complete the following steps:

1. Open the Manage Tivoli Enterprise Monitoring Services application.
2. Right-click on the service **Tivoli Enterprise Monitoring Server** and select **Reconfigure** from the menu.

3. On the TEMS configuration options window, enable the **TEC Event Integration Facility** check box.

4. Click **OK** twice. The TEC Server: Location and Port Number window opens.

5. In the TEC Server Location field, enter the host name where the EIF Probe was installed.

6. In the TEC port number field, enter the port number of the EIF Probe.

7. Click **OK** to close the window.

   Step 2: Edit the TEMS EIF configuration files

   The EIF configuration file requires the kd4.map file to work properly. Ensure the kd4.map file is copied to the directory where the EIF configuration file is located. The kd4.map file is available on the ITCAM for SOA V6.1 installation media.

   To edit the TEMS EIF configuration file:

1. On the IBM Tivoli Monitoring host, change to the directory where the TEMS is installed. The default directory is:
   Windows: \IBM\TM\cms\TECLIB\
   UNIX: <itm_installdir>/tables/host name/TECLIB/om_tec.config

2. Make a backp copy of the original OM_TEC.CONFIG file.

3. In a text editor open the file: OM_TEC.CONFIG.

4. Add the properties and values described in the following table:

Property Name	**Value**	**Description**
FilterMode	IN	Enables event filtering.
Filter:Class	ITM_Services_Inventory_610;	Event filters that forward all ITCAM for SOA events for the Services_Inventory_610 table to the Tivoli EIF Probe.
FilterCache:Class	ITM_Services_Inventory_610;	Filter that specifies which events to cache when IBM Tivoli Monitoring is unable to forward an event to the Tivoli EIF Probe

5. Save and close the file.

6. Restart the Tivoli Enterprise Monitoring Server

   Note: If the EIF configuration file already contains values for any of the parameters listed above, you should replace the current values with the values described in the table.

Step 3: Map situation events

The EIF probe maps the ITCAM for SOA situation events to Netcool/OMNIbus event tokens. The BSM_Identity strings created from these event tokens must match the BSM_Identity strings that were created from the**EventIdentifiers.xml** file for the Discovery Library toolkit.

By default, the situation events the EIF probe receives do not map all the required event tokens needed to identify the ITCAM for SOA services. The key event tokens excluded are:

- operation_name_u
- operation_namespace_u
- service_name_u
- port_namespace_u
- service_type

These event tokens uniquely identify a specific Service Port/Operation as defined by ITCAM for SOA. To map these event tokens, modify the tivoli_eif.rules file to build a BSM_Identity event token that maps to a specific TBSM service instance (in this case, an SOA resource). You make the changes in two different places in the tivoli_eif.rules file.
On Windows systems, the rules files is in the **%NCHOME%\omnibus\probes\win32** directory.
On UNIX systems, the path is **$NCHOME/probes/<os_name>**.

Before you begin, you need to Forward TEMS events to the EIF probe.

Customizing event token mapping

In this example, you create a logical block to remove single quotes from custom SOA event tokens. In this block all the events tokens used by the probe are stripped of single quotes which is required to avoid problems in later processing.

You also create an SOA-specific BSM_Identity token.

To configure the EIF Probes rule file with custom event tokens:

1. Open the **tivoli_eif.rules** file in a text editor.
2. Find the line with the text: **#End removal of embedded single quotes**.
3. To remove the single quotes from the custom SOA event tokens, enter the following code before the **#End removal of embedded single quotes** comments.

```
if(exists($operation_name_u ))
 {
                        if(regmatch($operation_name_u , "^'.*'$"))
                        {
                                $operation_name_u = extract($operation_name_u ,
"^'(.*)'$")
                        }
 }
 if(exists($operation_namespace_u ))
 {
                      if(regmatch($operation_namespace_u , "^'.*'$"))
                      {
 $operation_namespace_u = extract($operation_namespace_u , "^'(.*)'$")
                      }
 }
 if(exists($port_namespace_u ))
 {
                      if(regmatch($port_namespace_u , "^'.*'$"))
                      {
                              $port_namespace_u = extract($port_namespace_u ,
"^'(.*)'$")
                      }
 }

 '''if(exists($service_name_u ))'''
 {
                      '''if(regmatch($service_name_u , "^'.*'$"))'''
                      {
                              '''$service_name_u = extract($service_name_u ,
"^'(.*)'$")'''
                      }
 }
 if(exists($service_type ))
 {
                      if(regmatch($service_type , "^'.*'$"))
                      {
                              $service_type = extract($service_type , "^'(.*)'$")
                      }
 }
```

The quotes are stripped from the new SOA event tokens the same way they are for the default event tokens. The code identifies the specific event token (such as **$service_name_u**), and applies functions (regmatch and extract) and regular expression to the token to remove the single quotes.

4. To create the BSM_Identify field value from the custom SOA event tokens, enter the following code between the last two curly brackets at the end of the rules file:
   In this sample, the BSM_Identity field is build from the values for the operation_name_u, operation_namespace_u, service_name_u, and port_namespace_u SOA event tokens. **Note:** ITCAM for SOA sends a situation event for both the requestor and provider of the service port/operation pair. For this integration purpose you only filter events/situations the provider side by adding a check in the rules file for service_type=1.

5. Save and close the file.

6. To enable the changes, stop and restart the EIF Probe. In Windows, start the Tivoli_EIF probe service. In UNIX, execute the nco_p_tivoli_eif script from the **$NCHOME/omnibus/probes** directory.

Step 4: Customizing the Discovery Library toolkit

The Discovery Library toolkit filters ITCAM for SOA data and TBSM creates service instances based on the SOA data. The data can come from a DLA book file or a feed from TADDM host. To correctly manage the ITCAM for SOA data, you need to modify these XML control files on the TBSM host. These files are located in the following directory:

$NCHOME/XMLToolkit/xml

Event Identifier rules file

The **EventIdentifierRules.xml** file in the Discovery Library toolkit creates TBSM Identification field/value pairs for services imported from TADDM or DLA books. TBSM uses these identification values to correlate events to service instances. The default event identification rules provided with the Discovery Library toolkit map the IP Address or IBM Tivoli Managed managed system name to BSM_Identity values. To use the sample the integration with ITCAM for SOA, you need to customize these files.

Before you customize the event identification rule, backup the existing **EventIdentifierRules.xml** file. To handle the SOA data, you create new policy in the **EventIdentifierRules.xml** file. This Netcool/Impact policy defines an event identifier rule to match the status events from the SOA resource that were customized in the EIF Probe rules file.

1. In a text editor, open the **EventIdentifierRules.xml** file.
2. Search for the line: **<Mapping policy="GetAllIPAddressesAndFQDN" class="cdm:sys.OperatingSystem"/>**
3. Above this line, add the following code for the SOA event identifier policy:
   In the above policy for class type **WSOperation**, the event identification is built from a combination of the web services operation name, operation name space, port name and port name space.
4. Save and close the file.

   The above policy creates a TBSM BSM_Identification field value such as:

   As a result, each TBSM service instance created from the SOA data has a unique BSM_Identification field value. Whenever TBSM detects an event with this BSM_Identification field value, it will check the event for status information for the associated service.

   In this example, the event identifier policy creates BSM identifier with the operation name and port name.

   Using the above policy the BSM_Identification value is:

Labeling rules file

By default, TBSM creates the display name of the service instance from cdm:Label CDM attribute of the object. You can customize the **LabelingRules.xml** file to create the display name from other CDM attributes of the SOA object.

When you integrate TBSM with ITCAM for SOA, the service instance that was created is the name of the web service operation on a particular web service port. In TBSM, this value is the service display name. To maintain the uniqueness, you customize the name label to be:

```
web service operation name @ web service port
```

You also include the name and namespace attributes of both the objects.

Backup the existing **LabelingRules.xml** file and search for: <Mapping policy="Global" class="%" />

Add the policy in this code sample:

By performing the above customization, displayname of the service instance is generated as:

Composite rules file

The **CDM_TO_TBSM4x_MAP.xml** file groups a set of related objects into a composite representation of a single object. In some cases, the level of detail in the CCMDB is too granular. The composites create groups of objects to the level of management appropriate to the application. You can modify the default behavior of the CDM relationship types to a TBSM dependency to avoid the cyclical relationship between the various web services operations, web services ports and web services port end points.

Note: You need to have an in-depth understanding of the elements in the IBM Common Data Model before you attempt to change the composites configuration.

Composites are needed to group a set of related objects into a workable representation of a single object. If the level of detail in the CCMDB is too granular for certain applications, composites are used to create groups of objects at the level of management appropriate to the application. The default behavior of the CDM relationship types to a TBSM dependency was modified to avoid the cyclical relationship between the various web services operations, web services ports, and web services port end points. This is achieved by editing **CDM_TO_TBSM4x_MAP.xml** file in **%NCHOME%\XMLtoolkit\xml** directory.

Before you customize the file, backup the existing **CDM_TO_TBSM4x_MAP.xml** file. Search for following the string:

Add the lines from the following sample:

This change limits the relationship of cdm:invokedThrough has no dependency on target cdm.soa.WSPort on source cdm:soa.WSOperation. Similarly it also limits the dependency of cdm:soa.WSPort on cdm:soa.WSEndpoint.

Step 5: Processing the ITCAM for SOA DLA

The ITCAM for SOA 6.1 product shipped an ITCAM for SOA DLA that discovers SOA resources from your monitored environment. Follow the readme to discover services in your test or production environment. Run the ITCAM for SOA DLA following the documentation in the ITCAM for SOA DLA readme to generate an xml book with SOA resources in your monitored environment. A sample book (**ITCAMSOA61.cvtwin02.2007-05-01T02.13.52Z.refresh.xml**) discovered on our test environment is included in the TBSM_ITCAM4SOA.zip file shipped with the ITCAM for SOA Status Event Integration white paper at the TBSM Open Process Automation Library (OPAL).

There are two ways to process the IDML book:

- TBSM 4.1 Discovery Library Toolkit directly processing the IDML book generated by the SOA DLA
- Load the book into TADDM and TBSM 4.1 Discovery Library Toolkit integrates with TADDM and gets the information from TADDM

Process the DLA book

Copy the IDML book generated in the section above to **%NCHOME%\dlbooks** directory on TBSM Server which is the discovery library file system. Ensure the discovery library toolkit service is started. The discovery library toolkit service consumes the idml book from ITCAM for SOA and populates the service component registry with SOA resources.

The processing of the IDML book can be validated by checking the **msgGTM_XT.log** file for the message similar to the following:

```
GTMCL5290I: Book ITCAMSOA61.cvtwin02.2007-03-09T23.24.54Z.refresh.xml processed
successfully.
```

The following path indicates the default location of the log file:

- **Windows:** %NCHOME%\XMLtoolkit\log
- **UNIX:** $NCHOME/XMLtoolkit/log

Step 6: Create the TBSM service instance

Once the book is processed and the service component registry (SCR) is populated, the objects created in the SCR can be used to construct a business service in TBSM depending upon the requirement of service model.

Complete the following steps to create a service instance:

1. Login to TBSM console as a user with service administration permissions.
2. Select **Service Administration** from the Pages list and click **Go**.
3. Click **Templates** tab and click **Create New Template**.The service template describes a class of services. When you create the new service instance, you assign to this new template.
4. Give the name of the template as **ACMEServicesTemplate**.
5. In the Rules tab, click **Create a dependency rule**. This launches a dependency rule editor.
6. Name the dependency rule as **BSM_SOAOperationPortsWorst** with a Child Rule/Mapping as BSM_SOAOperationPort and click **OK**.
7. Save the new template.
8. Next, create a business service and assign it to the new service template. To add the services from SCR, click the **Dependents** tab and are click the **Add from Service Component Registry** button.
9. Select the SCR services created from the SOA DLA book.
10. When you have finished adding services, save the service template.

Step 7: Firing situation events

ITCAM for SOA has predefined situations that can be used to monitor service performance metrics. Use the Response Time Critical situations to illustrate the integration between TBSM and ITCAM for SOA. The Response Time Critical situation has a pre-defined value of 10 seconds. This means that when response to a web service exceeds 10 seconds a Response Time Critical situation will be fired by the IBM Tivoli monitoring infrastructure.

To trigger the situation, invoke the echoWithDelay operation using the web services sample application user interface with any string and a delay of greater than 10 seconds.

By default the sampling interval is five minutes, so one request with a response of 20 seconds in a five minute period should be enough to fire the event providing you make no other shorter requests during the same interval. This situation will show up as shown below in the Tivoli Enterprise Portal console.

When the response time critical situation fires in ITCAM for SOA an EIF event is sent to the Tivoli EIF probe which forwards that event to the OMNIbus server. This event is processed by the server and the SOA resource's status changes to RED because it is a critical situation that fired.

The following is the screen shot on the TBSM Service Tree:

Once the situation from ITCAM for SOA is cleared, ITM sends a clearing situation to TBSM and the ACME Business Service that is shown as red turns back to green. There are some additional customizations that can be done for the look and feel aspect of this integration to customize the flyover text of service instances in TBSM.

Tivoli Integrated Portal integration

Using a Tivoli Business Service Management Solution to Monitor PeopleSoft Environments

w3-connections.ibm.com

Integrating TBSM with Tivoli Application Dependency Discovery Manager

105-133 minutes

This paper is an update to the original IBM Tivoli Business Services Manager (TBSM) Integration With IBM Tivoli Application Dependency Discovery Manager (TADDM) white paper, "TBSM Integration with TADDM, Document version 1.0", published in December 2007. The focus of this paper is still the integration between TBSM and TADDM, but it reflects the changes that have been implemented in TBSM 4.2/4.2.1 and TADDM 7.1.2/7.2 with respect to the integration between the two products.

This integration involves the importing of data from TADDM into TBSM. The data import includes TADDM Business Applications and Business Services, which will be added to TBSM's list of services, and also the physical TADDM CI's, which will be added to TBSM's Service Component Repository (SCR). The SCR can be thought of as a physical list of resources that are available to be added to TBSM services. A key aspect of the SCR is the relationship information that is imported from TADDM, when an SCR resource is added to a service, not only the resource but also resources with relationships to that resource will be associated with the service.

One of the functions of the TBSM Discovery Library toolkit is to transition data from the IBM Common Data Model (CDM) into the template-based model used by TBSM. Both TADDM and the Discovery Library Adapters (DLA) represent data in the CDM. There have been a number of papers that have focused on the integration of TBSM with DLAs from other products, including Tivoli Monitoring and IBM Tivoli Composite Application Manager (ITCAM) for SOA. These papers are listed in the references below, and you are urged to also read these papers in order to get a complete understanding of the features available in the toolkit's CDM support. These papers have covered the customization available in the TBSM Discovery Library toolkit. This paper will include customization, but focus more on the mechanics involved in the TADDM integration and on issues that have come up as customers have worked to implement this integration.

The features mentioned in this paper are based on TBSM Discovery Library toolkit 4.2/4.2.1 and TADDM 7.1.2/7.2. Earlier levels of these products may not include all of the functionality mentioned in this paper.

Introduction

The TBSM Discovery Library toolkit acts as a bridge between the IBM Common Data Model (CDM) and the template-based model of TBSM. The toolkit can either interface with Tivoli Application Dependency Discovery Manager (TADDM) or read Discovery Library Adapter (DLA) books directly. This paper will primarily focus on the toolkit's interface with TADDM, although many of the principles discussed here are applicable to DLA books also. The toolkit consists of a process that collects information defined by the CDM and through a series of transforms, stores the data in the TBSM Service Component Repository (SCR). The SCR acts as a data source to TBSM, where an ESDA is used to extract data from the SCR and place the data into the TBSM server's data store.

The toolkit can either automatically poll TADDM for new and changed resources or it can wait until directed by a command line interface (CLI) to request data. Requests can be either bulk or delta. A bulk request asks TADDM for all of the objects of interest to TBSM. The toolkit will then update the SCR to match this data. An example of this is, if previously the SCR had objects A, B, and C from TADDM, and a new bulk request returned objects A, B, and D, then the SCR updates objects A and B, object C is deleted and object D is created. A delta request is simply a request to get any created, deleted, or modified objects since the last time an import was run.

The toolkit requests objects from TADDM on an object class basis. Filtering determines which object classes are imported. Requests interface with TADDM through TADDM's Java API. Jar files that are maintained on both the TADDM server and the TBSM server provide this API.

The returned data is further filtered and then passed through a series of transforms. The transforms are XSLT based, and use a series of XML files to take the CDM modeled data received from TADDM, and transforms these objects into objects and relationships consumable by the template based TBSM server. The XML files used by these transforms can be customized to better suit the needs of the customer.

Imported resources from TADDM to TBSM

The data stored in the SCR is pulled into the TBSM server through an external service dependency adaptor (ESDA). This ESDA will populate the Service Component Repository tab in the TBSM user interface. All "physical" resources received from TADDM are placed in the Service Component Repository. As an example, the following TADDM screen capture shows several Apache Web Servers.

These Apache Web Servers can be found in TBSM's Service Component Repository under the Application Servers, HTTP Servers.

Imported Business Application and Business Services from TADDM to TBSM

Business Applications and Business Services defined in TADDM will also populate the TBSM service tree. In the example below, TADDM contains the business application "Order Management".

This business application is placed in the TBSM Service tree under the Imported Business Services.

All Business Applications and Business Services, plus their contained/dependent children, imported from TADDM are placed under the Imported Business Services folder. When a physical object from the SCR is added to a service in TBSM, objects associated to that object via relationships provided by TADDM will also be added to the service. Relationships are followed, pulling in dependent objects until additional relationships that are interpreted as dependency relationships (see the CDM_TO_TBSM41x_MAP.xml file) are not found. Also, when the resources from TADDM are in the **Services** tab, they are available for event matching.

When objects imported from TADDM are placed in the SCR, the toolkit tracks which objects will affect the TBSM Services tree, and the toolkit automatically invalidates the upper-most affected object in the Imported Business Service. This invalidation will cause the ESDA to run and to update the services automatically. When the Service Navigation is refreshed, the new resources will be visible.

When TADDM imported resources are not part of a service, these resources are only visible in the **Service Component Repository** tab of the Service Navigation. The SCR tab is not invalidated automatically. To invalidate the SCR, select the Component Registry and invalidate, and then refresh the Service Navigation.

Define business services in TBSM with resource from TADDM

Physical resources can be added to a TBSM service by selecting the **Add from Service Component Repository** button on a service's dependents "Edit Service" panel:

Selecting the **Add from Service Component Repository** button opens a dialog box that contains the Component Registry tree. The tree can be expanded and the required resources selected.

The resources selected from the **Add from Service Component Registry Dialog** are added to the "Selected Services" area of the "Edit Service" dialog box. After the service is saved, the additional resources are displayed in the Service Navigation tree.

CDM to templates mapping

The TBSM Discovery Library toolkit installs a set of templates and mappings that will transform the more common collections of objects in TADDM to objects acceptable by TBSM. This includes computer systems, application servers, databases, and clusters to name a few. These templates provide a starting point, but it is expected that they will need to be customized and that additional templates created to meet a customer's unique needs. It also includes a set of event mappings that will map Tivoli Monitoring and z/OS® events to resources that have been loaded into TADDM via the TMS and z/OS DLA's.

The out of the box configuration of the toolkit does not provide a definitive solution for everyone. The verboseness of the CDM and the infinite number of event sources prevents a solution that covers all cases for all customers. The customization in the filtering and the XML transforms allows for the solution to be modified, meeting the specific needs of an organization.

Configuration

Supported Configurations

The TBSM Discovery Library toolkit acts as a bridge between the TADDM server and the TBSM server. The following tables list the supported configurations involving these components.

	TBSM 4.1.1 Server	**TBSM 4.2 Server**	**TBSM 4.2.1 Server**
TBSM 4.1.1 toolkit	YES	NO	NO
TBSM 4.2 toolkit	YES	YES	NO
TBSM 4.2.1 toolkit	YES	YES	YES

Table 1: TBSM Server / toolkit compatibility matrix

	TADDM 7.1	**TADDM 7.1.2**	**TADDM 7.2**
TBSM 4.1.1 toolkit	YES	NO	NO
TBSM 4.2 toolkit	YES	YES	NO
TBSM 4.2.1 toolkit	YES	YES	YES (see note)

Table 2 - TADDM Server / toolkit compatibility matrix

Note

- The TBSM 4.2.1 toolkit requires TBSM 4.2.1 FP1 (toolkit portion only, not the data server portion of FP1) in-order to import data from TADDM 7.2.
- It is *strongly* recommended that the TADDM Server be TADDM 7.1.2 FP6 or higher or TADDM 7.2 FP1 or higher, plus APAR IZ73202.
- If you are currently running with a TBSM 4.1.1 or 4.2 TBSM server, it is *strongly* recommended that you upgrade to the TBSM 4.2.1 Discovery Library toolkit.

Installing the TBSM 4.2.1 Discovery Library Toolkit with TBSM 4.1.1

To ease migration, the TBSM 4.2.1 Discovery Library toolkit can be installed on a TBSM 4.1.1. Server. This toolkit can interact with TADDM 7.1 or later systems.
Before installing the toolkit on a TBSM 4.1.1 Server, *back up your TBSM PostgreSQL database*. The installation will migrate the database schema used by the toolkit to the 4.2.1 database schema. The *IBM Tivoli Business Service Manager 4.2.1 Administrator's Guide*, Administering IBM Tivoli Business System Manager, discusses how to back up the database.

The TBSM 4.2.1 Discovery Library Toolkit is a new installation of the toolkit, not an upgrade to the currently installed version. This is due to directory structure changes in the TBSM 4.2/4.2.1 Server. Therefore the 4.2.1 toolkit will install into a different directory than your current 4.1.1 toolkit. Do not instruct the installer to use the same installation directory as your current 4.1.1 toolkit.

The toolkit's installer will detect the presence of a TBSM 4.1.1 toolkit and ask if you want to migrate; reply **yes**. By replying **yes**, the installer will copy the filter, configuration, and TADDM jar files from your older installation and migrate the database schema. The database schema migration can be lengthy, so be patient during this installation step.

Database Schema Migration

If you replied **no** when the installer asked if you wanted to migrate from 4.1.1, then you will need to run the ".../XMLtoolkit/bin/migrate_db_schema_to_42" script after the installation. This script will migrate the database. Depending on the size or your database, this process can be lengthy.
Issue "./migrate_db_schema_to_42.sh -?" for syntax. Here's a sample invocation:

```
./migrate_db_schema_to_42.sh -U postgres -d rad -p 5435 -l $TBSM_HOME/XMLtoolkit
```

Note: the "migrate_db_schema_to_42" might issue a message referring to the scc_systemconfig table. This message can be ignored; it does not indicate an error.

XML Customization File Migration

After installing the 4.2.1 toolkit, you will need to manually merge any changes that you have made to the following XML configuration files:

- CDM_TO_TBSM4x_MAP.xml
- CDM_TO_TBSM4x_MAP_Templates.xml
- EventIdentifierRules.xml
- LabelingRules.xml

If you specified **yes** to migrate from 4.1.1, then the 4.1.1 version of these files will be in $TBSM_HOME/XMLtoolkit/xml. The 4.2.1 version of these files will be in $TBSM_HOME/XMLtoolkit/xml/install. After you have merged these files, put the updated version in $TBSM_HOME/XMLtoolkit/xml.

TBSM

If the TBSM Discovery Library toolkit was installed with TADDM as the data source, then some of the items below will already be set for you. But it is still good to review each section, as some of these items have caused a few problems in the past.

Note: If you are running TBSM Discovery Library toolkit 4.2.1 with TADDM 7.1, several additional required configuration steps exist; these steps can be found in the Appendix.

Required Configuration Steps

API JAR Files

The information that follows applies to TADDM 7.1.2 or later. For jar file information for earlier releases of TADDM, please see the Appendix.
The TBSM Discovery Library toolkit uses TADDM's Java API to retrieve data from TADDM. This API requires that the API client jar file used by TBSM be the same as the version used by the TADDM server. Therefore, before starting the toolkit, the file(s) must be copied from the TADDM server to the TBSM server.

- TADDM 7.1.2
- The file, **taddm-api-client.jar**, can be found on the TADDM server in $COLLATION_HOME/sdk/clientlib and should be placed on the TBSM server in $TBSM_HOME/XMLtoolkit/sdk/clientlib.

- TADDM 7.2
- The files, **taddm-api-client.jar** and **platform-model.jar**, can be found on the TADDM server in $COLLATION_HOME/sdk/lib and should be placed on the TBSM server in $TBSM_HOME/XMLtoolkit /sdk/clientlib AND in in $TBSM_HOME/XMLtoolkit/sdk/lib.

If the jar files has not been downloaded to the TBSM server, the toolkit will write the following message to the msgGTM_XT.log and terminate the service:
GTMCL5361E: TADDM has been chosen as a data source for the toolkit, but the TADDM SDK jar files were not found. If connecting to a TADDM later than 7.1, please copy the jar files noted above from the TADDM server to $TBSM_HOME/XMLtoolkit/sdk/clientlib. If connecting to TADDM 7.1 or earlier, please copy all of the TADDM SDK jar files to $TBSM_HOME/XMLtoolkit/sdk/lib. The TADDM SDK jar files can be found on the TADDM server under /opt/IBM/cmdb/dist/sdk. After updating the jar files, restart the toolkit.

At the beginning of each import, TBSM will compare the checksum of the TADDM API jar file that it is using against the checksum of the jar file that the TADDM server is using. If the checksum of the TADDM API jar file used by the toolkit is different than the jar file used by the TADDM server, a warning message will be displayed in the msgGTM_XT.log:

GTMCL5355W: The TADDM Java API jar file taddm-api-client.jar, used by TBSM to communicate with the TADDM server, has changed on the TADDM server. To prevent possible interruptions in service it is recommended that you update the file on the TBSM server.

This message is a warning, the toolkit will continue to operate, but the jar file should be resynchronized at the earliest convenient time. After the jar files have been resynchronized, the TBSM toolkit must be restarted to pickup the new jar file.

Not keeping the TADDM API client jar file on the TBSM server in synch with the TADDM server may result in random exceptions when data is copied from TADDM to TBSM. Exceptions include:

- Unmarshalling errors, which in a nutshell indicate that the client side cannot decipher what the server sent.

- RMI exceptions

- Connection failures

These exceptions are written to the $NCHOME/XMLtoolkit/log/msgGTM_XT.log file.

Optional Configuration Steps

Database

Both the TBSM server and the TBSM Discovery Library toolkit use the PostgreSQL database. In TBSM 4.2, the database is configured by default to run optimally for average size databases. Large databases might require configuration changes. This is discussed more thoroughly in the white paper, "Tivoli Business Service Manager 4.1 Tuning Recommendations". The database configuration properties that affect the toolkit are discussed in the following sections.

Note: If using the TBSM 4.2.1 Discovery Library toolkit with a TBSM 4.1.1 server, the following recommendations must be made to the specified files.

$TBSM_HOME/tbsm/etc/rad_dbconf

This file contains a number of server configuration items. On UNIX platforms, the database properties in this file override the values in $TBSM_HOME/pgsql8/data/postgresql.conf.

PG_BUFFER is the same as shared buffers in the postgresql.conf file and is multiplied by the number of database connections allowed. It represents the number of shared memory buffers used by the database server. Each buffer is 8192 bytes. On UNIX systems, this value is sensitive to the maximum allowed shared memory, therefore the system configuration may need to be modified to allow for additional shared memory. If there is insufficient shared memory, the TBSM server will not start. If this happens, the value should be lowered until the TBSM server starts.

PG_BUFFER is set to 2000 out of the box. This should be reasonable for average size databases. If shared memory allows it, this value should be increased to provide better performance for larger databases. Ideally, if dealing with large TADDM databases or z/OS resources, set this value to a minimum of 3000.

$TBSM_HOME/pgsql8/data/postgresql.conf

This is the primary configuration file for PostgreSQL. The properties listed below are set to the values shown. Depending on the size of your database, these numbers may need to be increased to further optimize database performance.

Property	Suggested Value	Description
shared_buffers	2000	Keep this value the same as PG_BUFFER in rad_dbconf
work_mem	16192	Specifies the amount of memory to be used by internal sort operations and hash tables before switching to temporary tables. The value is specified in kilobytes. Each sort or hash operation associated with a complex query is allowed to use as much memory as this value specifies before it starts to put data to temporary files. For large numbers of resources in the database, adjusting this number upwards will help with ESDA performance and the responsiveness of clicking on the plus sign to get at the SCR data. Note: The toolkit sets the work_mem size on each of its connections. This size is configured by the property DL_DBManager.conf.bulk.work_mem, found in xmltoolkitsvc.properties. The default value is 32768.

maintenance_work_mem	32768	Specifies the maximum amount of memory to be used for maintenance operations.
wal_buffers	64	Specifies the number of buffers WAL (write ahead log) can have. Sixty-four corresponds to about 512K of memory. On UNIX, this value can request system shared memory.
checkpoint_segments	48	Maximum distance between automatic WAL checkpoints, in log file segments. Each segment is 16M.
checkpoint_timeout	600	Maximum time between automatic WAL checkpoints, in seconds.
effective_cache_size	32768	Sets the planner's assumption about the effective size of the disk cache that is available to a single index scan.

Table 3 - postgresql.conf customization

Additional sources pertaining to PostgreSQL performance can be found on the web.

After making these database configuration changes, the TBSM Data Server, Console Server, and PostgreSQL database must be stopped and restarted.

UserID and Password

The TBSM Discovery Library toolkit installation will prompt for the TADDM user identifier and password. These values are encrypted and stored on disk. If either value is incorrect, a connection exception will be logged in the message log, $TBSM_HOME/XMLtoolkit/log/msgGTM_XT.log:

If the TADDM user identifier and password need to be reset, or entered for the first time if the toolkit was originally installed to read books, the setxmlaccess command is used. The command is documented in the TBSM Administrator Guide, and online help is available by entering: "setxmlaccess -?".

Properties

The TBSM toolkit service is controlled through the property file xmltoolkitsvc.properties. The properties are set during the toolkit installation. These can be changed manually by editing the property file. Changes to these properties require the toolkit to be stopped and restarted. The properties that pertain to the TADDM integration are shown in the following table:

Property	Default Value	Description
DL_TADDM_Connect	true/false	The default value is set during installation, depending on whether TADDM or books will be the data source. Specifying true instructs the toolkit to use TADDM as the data source.
DL_TADDM_HostName	<TADDM_hostname>	The hostname or IP Address of the TADDM server. If TADDM is setup as an enterprise deployment, then the enterprise server should be specified.
DL_TADDM_Port	9530	The TADDM RMI port.
DL_TADDM_GUI_Port	9430	The TADDM HTTP port. This is used for launch.

DL_TADDM_Retry_Limit	864000	The number of seconds that the toolkit will retry if a connection can not be established with the TADDM server. If the retry limit is exceeded, then the current operation is canceled. This can result in the entire delta or bulk being canceled.
DL_Xform_Use_BCP	true	Instructs the toolkit to use COPY/BCP to load data into the stage tables. This will improve performance. *BCP uses the COPY command which can be sensitive to special characters in the data. The toolkit checks for these special characters, but there is some risk. This is worth turning on, but monitor the log for SQL exceptions.* Specify true to enable this function.
DL_PollIntervalSeconds	3600	Specifies how often the toolkit will go to TADDM to check for updates. The value is specified in seconds. If a value less than one is specified, then the toolkit will only check when directed by the cmdbdiscovery command. *The default value is 3600 seconds, be we recommend that this value be increased to more closely reflect the customer's environment.*
DL_TADDM_Immediate_Poll	true	Specifies that upon starting, immediately check with TADDM to see if there are updates. Specify false to wait DL_PollIntervalSeconds before checking with TADDM for updates.
DL_TADDM_ExplicitRelationship	true	The explicit relationships will be generated when an import is run. This should always be set to true.
DL_TADDM_DiscoveryWait	120	The value is specified in minutes. Prior to requesting data from TADDM, the toolkit will check to see if a TADDM discovery is running, if one is running, the toolkit will wait for the discovery to finish. This specifies how long the toolkit will wait. If the time expires, then the current action is canceled, and the toolkit will wait DL_PollIntervalSeconds before another attempt.
DL_TADDM_Immediate_Poll	True	When the toolkit service is started, normal operation is for it to immediately request data

		from TADDM. By setting this property to "false", the service will instead wait DL_PollIntervalSeconds before making its first request for data.
DL_MSSBatch	40	Number of GUIDs collected before a request is made for MSS data.
DL_RelationshipBatch	20	Number of GUIDs collected before a request is made for relationship information.
DL_DBManager.conf.bulk.work_mem	32768	Sets the database work_mem for each JDBC connection. For larger databases, this setting should be increased to 49152 or 65536.

Table 4 - xmltoolkitsvc.properties associated with TADDM

TADDM

There are not many items on the TADDM server that are specific to the TBSM integration. Performance is the main objective with most of these items.

Required Configuration Steps

API JAR Files

Again, simply a reminder that if a TADDM fix-pack is installed, then the jar file(s) in $COLLATION_HOME/sdk must be updated on the TBSM server in $TBSM_HOME?XMLtoolkit/sdk/clientlib directory. We cannot stress enough how important this is. Please see the earlier list for a description of the required jar files.

Update tables.extra

If the TBSM toolkit is connecting to an Enterprise TADDM or eTADDM, change_history_table must be added to the file ../cmdb/dist/etc/sync/tables.extra. If change_history_table is not added to this file, then change history data is not updated in the eTADDM when synchronizing with a domain TADDM. The change history data is needed for delta imports.

Optional Configuration Steps

Database

The performance of the TADDM database is paramount in the overall performance of the TADDM/TBSM integration.
It is suggested that you review the *Architecture, Deployment, Scalability, and Performance Overview* paper noted in the references.
There are more performance tuning suggestions listed in TADDM Administrator's guide, chapter 5, "Populating the Database", section "Performance tips for database.

Runtime

The toolkit is designed to run as a background process. In a nutshell, you start it, it waits for a specified interval to

expire, at which time it queries TADDM for updates, manipulates the data, and puts the data into the database. This interval can be interrupted via command line if an import is needed immediately.
Once configured, start the process. On Windows, the process is started either from the Windows Services Applet or via the net use command. On UNIX, the process is started using the tbsmrdr_start.sh script.

TBSM will perform two types of imports: bulk and delta.

- A bulk import means that TBSM will request all of the resources of interest from TADDM, and then synchronize the SCR with the current set of resources received from TADDM. As an example, lets say that prior to the bulk TBSM had resources "a", "b", and "c". A bulk import is started and TBSM gets resources "a", "b", and "d". TBSM would then update resources "a" and "b", delete resource "c", and create resource "d".

- A delta import means that TBSM will request all resources of interest in TADDM that changed between the last import and the current time. Changed resources include created, modified, and deleted resources. As an example, lets say that prior to the delta TBSM had resources "a", "b", and "c". A delta import is started and TBSM gets a delete for resource "b" and a create/modify for resources "a" and "d". TBSM would then delete resource "b", update resource "a", and create resource "d".

Note: TBSM processes all delta deletes in a batch, followed by all of the creates/modifies. TBSM currently does not have a means of accurately determining the exact time of the modification. Because of this restriction, if a TADDM operator creates a business application with resources "a" and "b", then removes resource "b" from the business application, and then a TBSM import is run. TBSM will create the service with resources "a" and "b" because it cannot distinguish the order of the delete and create of resource "b". To resolve this inconsistency, a TBSM bulk import will need to be run. If a delta import were run after the initial create of the business application, then TBSM would have created the service with resources "a" and "b". If a second delta import were run after resource "b" was removed from the business application, then TBSM would have deleted "b" from the service. This scenario would have avoided the need to run a bulk import.
The toolkit will check with TADDM for updates as defined by the polling interval (DL_PollIntervalSeconds) specified in the properties file. If updates are detected, then the toolkit will initiate a delta import and pickup the changes, after which it will sleep again until either the polling interval expires or a CLI (cmdbdiscovery) request is made.

The CLI (cmdbdiscovery) provides a means of initiating a bulk or delta import regardless of the polling interval. The complete usage of cmdbdiscovery can be seen by issuing "cmdbdiscovery -?", but we will discuss a couple of the more import flags here:

- -b - start a bulk
- -r - initiate import immediately
- -e - rebuild all explicit relationships

When the polling interval expires, the natural tendency of the toolkit is to run a delta import. By issuing "cmdbdiscovery -b", this tells the toolkit to run a bulk import the next time that the polling interval expires.

If the polling interval is set to one hour, but an import is needed now, then the -r flag should be use. Issuing "cmdbdiscovery -r" will result in a delta import starting. Issuing "cmdbdiscovery -b -r" will result in a bulk import starting. When the -r flag is used, the polling interval is stopped, and an import is started within sixty seconds. When the import is finished, the clock on the polling interval is reset back to zero and restarted.
The -e flag allows for the explicit relationships in the TADDM database to be rebuilt. The explicit relationships are built outside of the normal TADDM discovery process. When an import is requested by TBSM, TADDM will build any new explicit relationships based on resource changes since the last import. If it is suspected that the relationship table in TADDM is inaccurate, then issuing "cmdbdiscovery -b -r -e" will request that a bulk import with a complete rebuild of the relationship table be run. Depending on the size of the TADDM database,

rebuilding of the relationship table could be lengthy and it could impact TADDM performance. This should only be used when a problem is suspected, and the TADDM administrator should be notified beforehand.

To stop the process, on Windows the process is stopped either from the Windows Services Applet or via the net use command. On UNIX, the process is stopped using the tbsmrdr_stop.sh script.

Filtering

Out of the box, the TBSM toolkit imports a set of resources from TADDM that includes: computer systems, servers, databases, application servers, clusters, and business applications. This is a subset of what can be discovered by TADDM, and essentially any of the resources in TADDM can be imported into TBSM as long as the associated CDM classes are mapped to templates in TBSM. Exactly which resources in TADDM are of interest to TBSM and the business services that will be built and monitored in TBSM will vary with each implementation. The TBSM toolkit provides a number of filtering mechanisms, some minimize the amount of data that is requested from TADDM, others take the data that has been retrieved and reduces this further before writing it to the SCR database. The filters include:

- TADDM base class filters
- TADDM leaf node class filters
- Attribute global filters
- Attribute class filters
- View attribute filters (database view)
- Filtering TADDM Business Applications based on Attribute Values

One point should be made here, while any of the objects in TADDM can be imported into TBSM, and it may look nice to have very granular objects in some cases, if an event cannot be posted against an object, it may not make sense to show that object in TBSM. So a little common sense should be used when deciding which classes of objects to import into TBSM.

The graphic below illustrates the point in the processing when a particular filter is applied and the configuration files associated with each of the filters.

The filter files are located in $TBSM_HOME/XMLtoolkit/config/filters. The database is defined in $TBSM_HOME/XMLtoolkit/sql/scc_schema_views.sql.
The original approach taken was to filter very little. As time has passed, and we've seen how this approach has worked in the field, our view on filtering has changed, and we now believe that the initial approach should be to filter out most data initially, and then import additional classes as different types of information are needed in TBSM. The default filters provided with TBSM 4.2 are more restrictive than those in TBSM 4.1.1. There is additional discussion on this topic in the section on base class filters.

TADDM Model

Understanding which classes are in TADDM and what attributes are associated with each class can be a bit overwhelming given the granularity of the CDM. To help with this, the toolkit writes a number of files to the $TBSM_HOME/XMLtoolkit/config/cdmmeta directory:

- **BaseClassCMDB.xml** - this is simply a list of all of the TADDM base classes. A base class in this case is defined as a class that does not have a parent class defined in the model object returned by the TADDM metadata request. A sample entry in BaseClassCMDB.list is:

This entry is the name of the oracle.OracleServer class.

- **BaseClassCompleteCMDB.xml** - this file lists all of the classes and their parent class. Each line in the file lists a parent class, followed by the child class associated with that base class. This file looks very similar to the file, classfilters.xml, used for class filtering. A sample line in this file follows (to make the example readable, the class names are shortened):

  This line indicates that the class WebLogicEJBContainer is a child of the EJBContainer class.

- **<baseclass>.attrlist** - there will be one attrlist file for each class listed in BaseClassCMDB.list plus any class that is imported. Each attrlist file lists the attributes available for that class. Because an attribute is listed in this file, this does not mean that an instance of this class will have that attribute. This file is simply a list of the attributes that may appear in an object of this class or its subclasses and should only be used as a reference. The actual lists of attributes on an object is dependent on the data gathered by TADDM and the DLA books imported into TADDM. Each line in an attrlist file consists of the attribute name and the attribute type. An attribute type can be:

- a simple type, such as an integer
- a Java class, such as java.lang.String
- a TADDM object, such as com.collation.platform.model.topology.sys.Operating System

When TBSM receives an object from TADDM, it will request all attributes for that object. TBSM will process all attributes that are of type: string, integer, float, boolean, GUID, and long. Attributes defined as other types are discarded.

TBSM does not request TADDM extended attributes.

Each of these files is updated each time that the toolkit is started.

Class Filters

All class filtering is maintained in a single file, $TBSM_HOME/config/filters/classfilters.xml. This is a change from the previous release, in which separate files were kept for base class and child class filters.

Before we get into how the file works, the default class filtering provided with the earlier versions of the product was built with the premise that providing more than may be necessary was better than not providing enough. We have found that it is probably better to start with a small set and then add in classes that you find you need. With this in mind, the following classes are no longer imported by default:

Line	Description
com.collation.platform.model.topology.dev.MediaAccessDevice	objects representing motherboards, PCI adapters, memory, etc
com.collation.platform.model.topology.sys.SoftwareComponent	objects representing non-middleware components.
com.collation.platform.model.topology.dev.StorageExtent	disk partitions and storage volumes
com.collation.platform.model.topology.sys.FileSystem	local and remote file systems. The remote file systems can later be added by allowing

	this class and filtering the local file system classes via the child class filtering.
com.collation.platform.model.topology.net.LogicalConnection	Object representing logical connections between objects.

Table 5 - classfilters.xml default changes

The classfilters.xml file specifies which classes of resources that TBSM will import from TADDM. This file shows each base class and its children. Filtering can be specified at either the base class level or at the leaf node level. Filter specification of intermediate nodes is ignored; this will be explained as we go.
The example below will be used to describe how the filters work. To make the example more readable, the class names have been shortened for this example.

Figure 10 - Sample classfilter.xml Segment

Each line is either a <baseclass> or a <childclass>, and contains the attributes:

- import - true indicates this is a class of resource that should be imported. False indicates this class of resource should not be imported.
- class - the name of the class
- depth - the depth in the tree, starting with the base class which has a depth of 0. Do not confuse this with the depth flag on the TADDM api.sh script, they are not related.

The classfilters.xml is basically a list of base classes, and contained within each base class is its children. The "<baseclass>" tag defines the base class. In the example above, "topology.sys.FileSystem" is the base class.

The base class "topology.sys.FileSystem" has eight children, each defined by the tag "<childclass>". The leaf nodes are the children at the lowest leg of each branch. In this example there are five leaf nodes, they are:

- topology.sys.openvms.OpenVmsFileSystem
- topology.sys.sun.SolarisFileSystem
- topology.sys.windows.WindowFileSystem
- topology.sys.NFSFileSystem
- topology.sys.SMBFileSystem

Now you might ask, "topology.sys.openvms.OpenVmsFileSystem" and "topology.sys.unix.UnixFileSystem" are at the same level in the tree, why is one a leaf node and the other not? Looking at the tree, "topology.sys.unix.UnixFileSystem" has a child, "topology.sys.sun.SolarisFileSystem", so the leaf node here is "topology.sys.sun.SolarisFileSystem", while "topology.sys.openvms.OpenVmsFileSystem" does not have any children so it is a leaf node. Using the depth attribute and the indentation should help in seeing this.

Now that the concepts are defined, this paper will discuss how the file works. Earlier it was mentioned that filters could be defined at the base class level and the leaf node level.

A child will inherit the import state of its base class. So setting import='false' for "topology.sys.FileSystem" means that it and all of its children have import='false', and therefore none of the objects in this class will be imported. Likewise, setting import='true' on "topology.sys.FileSystem" means that this base class and all of its children are imported.

Now assume that in general you are not interested in FileSystem objects, but you are interested in

RemoteFileSystems and you changed the previous example in the following way:

Figure 11 - Sample classfilters.xml leaf node filter

This example tells the system to import resources of type "topology.sys.NFSFileSystem" and "topology.sys.SMBFileSystem", but none of the other file system objects.

This change has two benefits:

- TBSM only imports the objects of interest
- It helps with performance because less information is requested from TADDM and the database only contains objects of interest.

Now you might ask, why could I not leave everything as import='false' and simply change the following line:

<childclass import='true' class='topology.sys.RemoteFileSystem' depth='1'/>

This type of change does not work because "topology.sys.RemoteFileSystem" is not a leaf node and therefore it inherits its import state from its base class.
To switch hats, say that you are interested in FileSystems with the exception of WindowsFileSystem. In this case you change the previous example to:

In this case TBSM will request all topology.sys.FileSystem objects, and then remove the topology.sys.windows.WindowsFileSystem objects when they are received. This will not reduce the overhead of pulling data over from TADDM, but it will reduce the number of objects in the TBSM database.

Lastly you might ask, since the toolkit is only looking at the import state for the base classes and the leaf nodes, does it really matter what the import value is for the intermediate child classes? Technically no, the import state of the intermediate child classes is ignored, but to avoid confusion set all of the import values for the child classes to the same value as the base class.

Global Attribute Filters

There is a set of attributes that in general are not of interest and therefore are not worth instantiating when a resource is written to the TBSM database. This set of attributes is defined in the file, $TBSM_HOME/XMLtoolkit/config/filgers/AttributeGlobal.xml. This file is customizable. The attributes specified in this file will be removed from all objects requested from TADDM.

This means that you will not see these attributes under the **Additional** tab in the service editor and these attributes cannot be used in any of the XML configuration files.

Looking in any of the *.attrlist files in the $NCHOME/XMLtoolkit/cdmmeta/attributes directory, you will see:

Each line in the *.attrlist files has the attribute name and the attribute type. The attribute value in the AttributeGlobal.xml file is simply the name of the attribute taken from the *.attrlist file.

Class Attribute Filters

For each class that is imported, an <classname>.attrlist file is generated. These files can be used to generate attribute filters at the class level. The name of these filters is <classname>.xml. The name of the filter for BindAddress is:

The format of the file is the same as the AttributeGlobal.xml file.

One shortfall here is that the <classname>.attrlist file only lists the attributes at the base class level, it does not contain any additional attributes for any classes that are children of the base class. To see any additional attributes for the child classes, you will need to either look at an object's additional attributes tab in the TBSM UI or look at the details of an object in the TADDM UI.

If a TADDM import has already been run, the following SQL will provide an understanding of the types of attributes that TBSM has received data for.
To see a list of all of the CDM classes that TBSM has received data for, execute the following SQL:

select distinct class from scc_components

To see the attributes that TBSM has received for a particular class, execute the following SQL:

where scc.class='<className>'

Where <className> is the class of interest (i.e. cdm:app.AppServer).

To see all of the attributes for a particular class, ordered by the class that they are associated with, execute the following SQL:

See the section about Analyzing the SQL Tables for an explanation on running SQL.

View Attribute Filter

The view attribute filter is an optional view into the attribute table that limits the data returned by the ESDA to only those attributes that are deemed necessary. The IN() clause on the SQL where statement acts as the filter. The example in Figure 17 limits the attributes to the eight shown:

The view definition is in the file $TBSM_HOME/XMLtoolkit/sql/scc_schema_views.sql. To change the definition, copy the file and edit the IN() clause to include the required attributes. Using psql or pgadmin, drop the current view

Then run the create view statement to recreate the view. The view definition for view_componentAttributesLimited includes two UNION statements so that the _sourceToken and _sourceContactInfo keywords are included, be sure to include these in your new view definition.

Filtering TADDM Business Applications by Attribute Value

A request was made to allow for additional filtering of TADDM business applications, allowing for a subset of those defined in TADDM to be imported into TBSM. As an interim change, a new property can be added to xmltoolkitsvc.properties, DL_LifeCycleState_Filter. This property will represent a "WHERE" clause on the request for business applications from TADDM.

As an example, if:

DL_LifeCycleState_Filter=WHERE (lifecycleState == 14) OR (lifecycleState == 20)

Then TBSM would issue:

SELECT * from ITSystem where (lifecycleState == 14) OR (lifecycleState == 20)

Resulting in only those business applications that have the lifecycleState attribute set to 14 or 20 are imported into TBSM. The attribute specified in the "where" clause must be an attribute that TADDM allows to be specified on a SQL query. This "where" clause will only be applied to the **ITSystem class**, which represents business applications in TADDM.

Before setting the DL_LifeCycleState_Filter, the "where" clause should be verified by running the request using the TADDM api.sh script. If an invalid attribute is specified in the "where" or if the case of the attribute's letters are incorrect, TADDM will throw an exception when the request is made. So it is very important to test the "where" clause using the api.sh script before setting the value in the xmltoolkitsvc.properties file.

The attributes associated with the TADDM ITSystem class can be found in the directory $NCHOME/XMLtoolkit /config, the file is: com.collation.platform.model.topology.sys.ITSystem.attrlist. This file will be generated the first time that the toolkit is started. Not all of the attributes listed in this file are allowed by TADDM on its SQL queries.

Class and Relationship Mapping

The data imported from TADDM abides by the CDM architecture. The common data model is verbose, defining hundreds of different objects and relationships. Meanwhile, status for these objects is provided by events received by NetCool/OMNIbus and other data sources. These events must be mapped to the objects imported from TADDM.
The TBSM Discovery Library toolkit populates the SCR, which acts as a bridging area between the resources and relationships defined by the CDM and the template-based service model used by TBSM. TBSM includes a default set of CDM objects mapped to TBSM templates. There is also a set of event identifiers that provide the mapping of events to the imported common data model objects.

To extend the default support, the toolkit allows for four areas of customization:

- Mapping - maps CDM classes and relationships to templates
- Composites - defines composites, the combination of several CDM objects into a single class, which is then mapped to a template.
- Event Identifiers - the definition of an identifier that identifies an object for purpose of aligning outside pieces of data (e.g. events, KPI) with the CDM object. The native identifier for a CDM object is the GUID or a naming string, neither of which is likely to appear in an event.
The term "event identifiers" is a bit of a misnomer. Think of them as resources, identifiers, or aliases used to map to events and other information about a resources provided by data fetchers. These identifiers are visible in the Identifiers Tab in the TBSM Service Editor.
- Labeling - defines the attribute or attributes to be used for the display name.

A detailed write-up of the imported process is in the *IBM Tivoli Business Service Manager Administrator's Guide*(see the "Discovery Library toolkit" chapter).

The Common Data Model

The common data model defines the data model employed by TADDM. The current model can be broken down into 16 sections. See the IBM Common Data Model [7] in the references for additional information on the CDM. This reference will direct you to the location of the CDM and diagrams and information discussed below.

You can drill down into each section of the overview, yielding the classes and relationships the section is composed of. Selecting **Computer System** yields the diagram:

Clearly a Microsoft Word document is not the optimal media to describe the CDM. But the preceding graphic highlights the point that the CDM is very detailed, much more detailed than what is required for BSM, and that only a subset of the data defined by the CDM is needed by TBSM.

Selecting any of the classes or relationships in the preceding diagram will display a textual description of the item selected. For a relationship, the information includes: classes using the relationship, directionality, and diagrams including this interface. For a class, the information includes: derivation hierarchy, attributes, interfaces implemented, relationships, naming rules, and diagrams including this class.

An in-depth discussion of the CDM is far beyond the scope of this paper. The points to be made here are that specific objects within the CDM are of interest when monitoring at the business system level. In some cases it may be as simple as a single object, in other cases it may be a collection of combined objects (a composite) that comprise a meaningful entity. These objects and their relationships with other objects can be imported into TBSM and used to build business services.

Class to Template Mapping

The data stored in TADDM is defined by the CDM. To get from the CDM to the template based structure in TBSM, the toolkit must map between the CDM classes that TBSM receives from TADDM, to the templates defined in TBSM, this mapping is defined by the file CDM_TO_TBSM4x_MAP_Templates.xml. This file does not create or define TBSM templates, it simply maps between the template and the CDM instance. The templates that are referenced are already be defined in TBSM.

The table that follows lists each of the CDM classes, the TADDM base class that they are contained in, and the TBSM template that uses this class, as defined by the CDM_TO_TBSM4x_MAP_Templates.xml that is included with TBSM. The CDM_TO_TBSM4x_MAP_Templates.xml file is a living document; refer to the actual file for the latest mappings.

Note: See $TBSM_HOME/XMLtoolkit/sql/scc_collect_db_info.sql for queries that show the class-to-template mappings.

CDM Class	**TADDM Base Class**	**TBSM Template**
sys.ComputerSystem	sys.ComputerSystem	BSM_Node
sys.GenericComputerSystem	sys.ComputerSystem	BSM_Node
sys.UnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.AixUnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.FreeBSDUnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.HpUnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.IPSOUnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.LinuxUnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.NetwareUnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.OpenVmsUnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.SunSPARCUnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.SunSPARCVirtualComputerSystem	sys.ComputerSystem	BSM_Node
sys.VmwareUnitaryComputerSystem	sys.ComputerSystem	BSM_Node
sys.WindowsComputerSystem	sys.ComputerSystem	BSM_Node
sys.zOS.ZVM	sys.OperatingSystem	BSM_Node
sys.zOS.Zlinux	sys.OperatingSystem	BSM_Node
sys.zOS.ZOS	sys.OperatingSystem	BSM_Node
sys.zOS.ZseriesComputerSystem	sys.ComputerSystem	BSM_Node_Z

app.web.apache.ApacheServer	App.AppServer	BSM_AppServer
app.AppServer	App.AppServer	BSM_AppServer
sys.zOS.AddressSpace	sys.RuntimeProcess	BSM_AppServer
app.GenericAppServer	App.AppServer	BSM_AppServer
app.Web.iplanet.IplanetProxyServer	App.AppServer	BSM_AppServer
app.lotus.DominoServer	App.AppServer	BSM_AppServer
app.j2ee.J2EEServer	App.AppServer	BSM_J2EEServer
app.j2ee.GenericJ2EEServer	App.AppServer	BSM_J2EEServer
app.j2ee.jboss.JbossServer	App.AppServer	BSM_J2EEServer
app.web.apache.ApacheServer	App.AppServer	BSM_HTTP
app.web.GenericWebServer	App.AppServer	BSM_HTTP
app.web.iis.IisWebService	App.AppServer	BSM_HTTP
app.web.WebServer	App.AppServer	BSM_HTTP
app.web.iplanet.IPlanetServer	App.AppServer	BSM_HTTP
app.web.ibm.IBMHTTPServer	App.AppServer	BSM_HTTP
sys.zOS.CICSRegion	App.AppServer	BSM_CICS
app. AppServerFunctionalGroup	Core.Collection	BSM_AppLogicalGrouping
app.SoftwareModuleFunctionalGroup	Core.Collection	BSM_AppLogicalGrouping
app.FunctionalGroup	Core.Collection	BSM_AppLogicalGrouping
core.Collection	Core.Collection	BSM_AppLogicalGrouping
app.j2ee.websphere.WebSphereCell	App.j2ee.J2EEDomain	BSM_AppServerGroup
app.j2ee.jboss.JBossDomain	App.j2ee.J2EEDomain	BSM_AppServerGroup
sys.zOS.MQSubsystem	App.AppServer	BSM_AppLogicalGrouping
sys.zOS.Sysplex	Core.Collection	BSM_ServerCluster
sys.zOS.SysplexGroup	Core.Collection	BSM_ServerCluster
sys.zOS.IMSSubsystem	App.AppServer	BSM_AppLogicalGrouping
app.j2ee.weblogic.WebLogicServer	App.AppServer	BSM_WeblogicServer
app.Application	sys.ITSystem	BSM_BusinessApplication
process.Activity	process.Activity	BSM_BusinessApplication
app.j2ee.J2EEApplication	Core.Collection	BSM_J2EEApplication

app.j2ee.jboss.JBossJ2EEApplication	Core.Collection	BSM_J2EEApplication
app.j2ee.weblogic.WebLogicJ2EEApplication	Core.Collection	BSM_J2EEApplication
app.j2ee.websphere.WebSphereJ2EEApplication	Core.Collection	BSM_J2EEApplication
app.j2ee.WebModule	Core.Collection	BSM_J2EEApplication
app.j2ee.EJBModule	Core.Collection	BSM_J2EEApplication
app.j2ee.jboss.JBossEJBModule	Core.Collection	BSM_J2EEApplication
app.j2ee.weblogic.WebLogicEJBModule	Core.Collection	BSM_J2EEApplication
app.j2ee.websphere.WebSphereEJBModule	Core.Collection	BSM_J2EEApplication
app.j2ee.jboss.JBossEJB	Core.Collection	BSM_J2EEApplication
app.j2ee.weblogic.WebLogicEJB		BSM_J2EEApplication
app.j2ee.websphere.WebSphereEJB	app.j2ee.J2EEComponent	BSM_J2EEApplication
app.j2ee.websphere.WebSphereWebModule	app.SoftwareModule	BSM_J2EEApplication
app.j2ee.weblogic.WebLogicWebModule	app.SoftwareModule	BSM_J2EEApplication
app.j2ee.jboss.JBossWebModule	app.SoftwareModule	BSM_J2EEApplication
app.web.apache.ApacheModule	app.SoftwareModule	BSM_J2EEApplication
app.web.iplanet.IPlanetJSP	app.SoftwareModule	BSM_J2EEApplication
app.web.iis.IIsModule	app.SoftwareModule	BSM_J2EEApplication
app.web.iplanet.NSAPIPlugin	app.SoftwareModule	BSM_J2EEApplication
app.web.iplanet.WebLogicPlugin	app.SoftwareModule	BSM_J2EEApplication
app.web.CGIScript	app.SoftwareModule	BSM_J2EEApplication
app.j2ee.jboss.JBossServlets	app.j2ee.J2EEComponent	BSM_J2EEApplication
soa.BusinessProcess process.BusinessProcess		BSM_SOAProcess
soa.WSOperation	process.Activity	BSM_SOAOperationPort
soa.WebService	sys.Service	BSM_SOAService
app.lotus.DominoApplication	app.lotus.DominoDatabase	BSM_Application
sys.BusinessSystem	sys.ITSystem	BSM_BusinessService
process.BusinessProcess	process.Activity	BSM_BusinessService
process.BusinessService	process.Activity	BSM_BusinessService
app.db.Database		BSM_Database

app.db.db2.Db2Database	app.db.db2.Db2Database	BSM_DB2Database
sys.zOS.DB2Subsystem	app.AppServer	BSM_DB2DatabaseServer
app.db.db2.Db2Server	app.AppServer	BSM_DB2DatabaseServer
app.db.db2.Db2System	app.db.db2.Db2System	BSM_DB2DatabaseServer
app.db.db2.Db2Instance	app.AppServer	BSM_DB2DatabaseServer
app.db.oracle.OracleDatabase	app.db.oracle.OracleDatabase	BSM_OracleDatabase
app.db.oracle.OracleInstance	app.AppServer	BSM_OracleDatabaseServer
app.db.oracle.OracleServer	app.db.oracle.OracleServer	BSM_OracleDatabaseServer
app.db.sybase.SybaseServer	app.AppServer	BSM_SybaseDatabaseServer
app.db.sybase.SybaseDatabase	app.db.sybase.SybaseDatabase	BSM_SybaseDatabase
app.db.mssql.SqlServerDatabase	app.db.mssql.SqlServerDatabase	BSM_MSSQLDatabase
app.db.mssql.SqlServer	app.AppServer	BSM_MSSQLDatabaseServer
app.j2ee.websphere.WebSphereServer	app.AppServer	BSM_WebSphereServer
tbsm:net.Router	<<composite>>	BSM_Router
tbsm:net.Switch	<<composite>>	BSM_Switch
tbsm:net.Bridge	<<composite>>	BSM_Bridge
tbsm:net.LoadBalancer	<<composite>>	BSM_LoadBalancer
Tbsm:net.Firewall	<<composite>>	BSM_Firewall
sys.NFSService	sys.Service	BSM_NFSService
sys.DNSService	sys.Service	BSM_DNSService
sys.LDAPService	sys.Service	BSM_LDAPService
app.AppServerCluster	app.AppServerCluster	BSM_AppServerCluster
app.lotus.DominoCluster	app.AppServerCluster	BSM_AppServerCluster
app.GenericAppServerCluster	app.AppServerCluster	BSM_AppServerCluster
app.j2ee.jboss.JBossCluster	app.AppServerCluster	BSM_AppServerCluster
app.j2ee.websphere.WebSphereCluster	app.AppServerCluster	BSM_WebsphereCluster
app.j2ee.weblogic.WebLogicCluster	app.AppServerCluster	BSM_WeblogicCluster
ActiveDirectory	sys.Service	BSM_Service
sys.zOS.DB2DataSharingGroup	core.collection	BSM_AppServerCluster
sys.zOS.IMSSysplexGroup	core.collection	BSM_AppServerCluster

Table 6 - CDM to TADDM class to TBSM template mappings

The syntax of the XML in this file is documented in the *IBM Tivoli Business Service Manager Customization Guide*. Using this information, additional classes can be mapped to either existing templates, or to newly created templates.

If resources visible in TADDM are not visible in TBSM after an import, the first item to check is which TADDM class the object belongs, and then check the CDM_TO_TBSM4x_MAP_Templates.xml file to see if the class is mapped to a template. If a class is not mapped, then it will not be visible in TBSM. The TADDM user interface displays a "user-friendly" version of the object type rather than the actual class name.

So a little bit of intuition needs to be used to map between the object type in the TADDM UI and the class name. In this case, if you search the file BaseClassCompleteCMDB.list for SPARC, you will see that the class name is "sys.sun.SunSPARCUnitaryComputerSystem", which has base class "sys.ComputerSystem".

Instance-Based Template Mapping

A new function for TBSM 4.2 is instance-based template mapping. The earlier section discussed mapping CDM classes to TBSM templates using the CDM_TO_TBSM4x_MAP_Templates.xml file. This mapping provides a class to template mapping and was available in previous releases of the toolkit. Instance-based template mapping allows conditional mapping of an instance of a CDM classes to a template. So rather than mapping all instances of a particular class to a particular template, you can now specify that only instances of a class with particular attributes should be mapped to a particular template. Examples of when instance-based template mapping might be used include:

- Specifying that a subset, rather than all, of the IBM Tivoli Monitoring collections should be created as business systems.
- Disabling an instance of a particular class from having a template.

Instance-based templates are defined in the CDM_TO_TBSM4x_MAP_Templates.xml file. They are specified outside of the <classmappings> portion of the document and use the same <Policy> and <Mapping> syntax that is used in the CDM_TO_TBSM4x_MAP.xml file to define composites.

Section 6.5 provides an example that includes instance-based template mapping. This example will provide a better understanding of the mechanics of instance-based template mapping.

Composites

In addition to mapping a specific CDM class to a TBSM template, a collection of CDM classes can be combined into a single object, referred to as a *composite*. This composite can then be mapped to a template. Composites included with TBSM are defined in the CDM_TO_TBSM4x_MAP.xml file.

The CDM_TO_TBSM4x_MAP.xml file contains policies that define the composite, and mappings that associate a particular CDM class to the policy.

This says that a TADDM resource of CDM type cdm:sys.zOS.Sysplex should be examined by policy SysplexComposite to see if the resources should be represented by a composite.

The policy definition defines the requirements needed to represent the resource as a composite.

The policy definition states that the resource must have a cdm:hostedCollection relationship with a resource of CDM class cdm:core.SystemSpecificCollection. If the original cdm:sys.zOS.Sysplex resource satisfies this, then the attributes and relationships from the two resources are combined, and represented by a single object that started the examination, in this case a sysplex.

Several composite definitions are included with TBSM, as listed in the following table.

The **TBSM Composite** column first lists the policy name that represents the composite. The **CDM Class** column for each composite is broken down into two sections. The first section lists the primary CDM class associated with the composite. The second section lists the secondary CDM classes that are pulled into the composite. The **TBSM Class** column lists the tbsm namespace objects created to represent this composite. This column is populated only if the composite creates a name space object.

The mappings in CDM_TO_TBSM4x_MAP_Templates.xml will contain a mapping for the classes in the TBSM Class column. If the TBSM Class column is blank, then there will be a mapping for the primary CDM class in the CDM Class column. The primary class is the first class listed. More advanced composite definitions are capable of creating a new object as a result of the policy that is driven by a CDM object. In the following table those policies that have a value in the **TBSM class** column are generating new objects.

TBSM Composite	**CDM Classes Contained in the Composite**	**TBSM Class**
TBSMRouter	sys.UnitaryComputerSyste, sys.ComputerSystem net.Router	tbsm:net.Router
TBSMSwitch	sys.ComputerSystem, net.L2Interface, net.Segment	tbsm:net.Switch
Nodes	sys.ComputerSystem, sys.GenericComputerSystem, sys.aix.AixUnitaryComputerSystem, sys.freebsd.FreeBSDUnitaryComputerSystem, sys.hpux.HpUnitaryComputerSystem, sys.ipso.IPSOUnitaryComputerSystem, sys.linux.LinuxUnitaryComputerSystem, sys.netware.NetwareUnitaryComputerSystem, sys.openvms.OpenVmsUnitaryComputerSystem, sys.sun.SunSPARCUnitaryComputerSystem, sys.sun.SunSPARCVirtualComputerSystem, sys.vmware.VmwareUnitaryComputerSystem, sys.windows.WindowsComputerSystem sys.sun.Solaris, sys.linux.Linux, sys.ipso.Ipso, sys.hpux.HpUx, sys.aix.Aix, sys.freebsd.FreeBSD, sys.netware.Netware, sys.openvms.OpenVms, sys.unix.Unix, sys.windows.WindowsOperatingSystem, sys.zOS.ZVM, sys.zOS.ZOS, sys.zOS.ZLinux. sys.vmware.VmwareESX, net.IpInterface, net.L2Interface, net.IpV4Address, net.IpV6Address, net.Fqdn	
Services	sys.DNSService, sys.NFSService, sys.LDAPService	
SysplexComposite	sys.zOS.Sysplex core.SystemSpecificCollection	
NetworkServices	net.Bridge	

WSOperationPort	soa.WSOperation soa.WSPort, soa.WSEndpoint
forCICSRegionGrps	
cdm:sys.zOS.CICSRegion cdm:sys.zOS.ZOS	tbsm:CICSFileGroup tbsm:CICSProgramGroup tbsm:CICSTransactionGroup
forMQGrps	
cdm:sys.zOS.MQSubsystem cdm:sys.zOS.ZOS	tbsm:MQSenderChannelGroup tbsm:MQReceiverChannelGroup
forIMSGrps	
cdm:sys.zOS.IMSSubsystem cdm:sys.zOS.ZOS cdm:sys.zOS.IMSSysplexGroup	tbsm:IMSProgramGroup tbsm:IMSTransactionGroup tbsm:IMSDatabaseGroup
forIMSSysplexGrps	cdm:sys.zOS.IMSSysplexGroup
tbsm:IMSProgramGroup tbsm:IMSTransactionGroup tbsm:IMSDatabaseGroup	
IMSTranCluster	cdm:sys.zOS.IMSSysplexGroup cdm:sys.zOS.IMSSubsystem cdm:sys.zOS.IMSTransaction
TBSMLoadBalancer	
sys.UnitaryComputerSystem, sys.ComputerSystem cdm:net.vip.VipFunction cdm:net.vip.bigip.BigIPVipFunction	tbsm:net.LoadBalancer
TBSMFirewall	
sys.UnitaryComputerSystem, sys.ComputerSystem cdm:net.vip.VipFunction cdm:net.vip.bigip.BigIPVipFunction	tbsm:net.Firewall
TBSMBridge	
sys.UnitaryComputerSystem, sys.ComputerSystemcdm:net.Bridge	tbsm:net.Bridge

Table 7 - TBSM Composite definitions and associated CDM and TADDM classes

This file is a living document; refer to the CDM_TO_TBSM4x_MAP.xml file for the latest composite definitions.

Relationships

The CDM defines not only the classes that comprise the model, but the relationships between these classes. Each relationship has a defined source and target class. At times, the source and target order in the model does not match the dependency model defined in TBSM. The toolkit allows for this interpretation to be reversed, either globally or specific to certain classes. The toolkit also allows for a relationship to not be treated as a dependency, which essentially causes the relationship to be ignored. These definitions are in the file CDM_TO_TBSM4x_MAP.xml.

The cdm:systemDependency relationship is an example of ignoring a relationship. This relationship often results in the ball of yarn effect on the topology, as everything tends to appear to be dependent on everything else. Therefore you want to ignore these relationships. Looking in CDM_TO_TBSM4x_MAP.xml you will see the following definition, which causes TBSM to ignore all cdm:systemDependency relationships.

The cdm:memberOf relationship is an example of a relationship that needs to be reversed. In TADDM an AppServer is a memberOf a FunctionalGroup. In TADDM a functionalGroup is a logical grouping. Without reversing the memberOf relationship in CDM_TO_TBSM4x_MAP.xml, the functionalGroup would appear under the AppServer, which is the opposite of what we need. The following definition has been added to CDM_TO_TBSM4x_MAP.xml so that TBSM reverses the direction of the cdm:memberOf relationship.

Template Mapping Example

The CDM_TO_TBSM4x_MAP_Templates.xml file contains several instance-based template mappings. As an example, one of the mappings is discussed in this section.
Outside of the <classmappings> portion of the document you will find the mapping policy and the mapping template.

Similar to the definition of a composite, instance-based template mapping uses <Mapping> tags to associate a CDM class with a policy. In this case, the cdm:sys.ComputerSystem and cdm:sys.UnitaryComputerSystem classes are associated with the "NotComputer" policy.

This is a good example because it will pull in both composite and template mappings. In the <classmappings> section of the CDM_TO_TBSM4x_MAP_Templates.xml document, you have the template mapping:

The mapping shown in Figure 24 indicates that both cdm:sys.ComputerSystem and cdm:sys.UnitaryComputerSystem will both be mapped to the BSM_Node template, in addition to the other classes listed in this <template> definition.

In TBSM 4.1.1 this would have been the end of the story, all objects of these classes would be a BSM_Node. This sounds simple enough, but there is a catch if you look in the CDM_TO_TBSM4x_MAP.xml document. This document contains the composite definitions, and in these definitions you will find mappings.These are mappings to build composites for some of the networking resources, and it indicates that the networking resources are also received as cdm:sys.ComputerSystem and cdm:sys.UnitaryComputerSystem.

For Router, the mapping indicates that a cdm:sys.ComputerSystem and a cdm:sys.UnitaryComputerSystem should be evaluated against the policy "TBSMRouter". This policy checks to see if the object has the attribute "cdm:Type" and whether it has the value "Router". It then checks to see if the object has a "cdm: Provides" relationship to a "cdm:net.Router" object. If these are true, then a composite is built for class "tbsm:net.Router".

If you look back in the CDM_TO_TBSM4x_MAP_Templates.xml document, you see that "tbsm:net.Router" is mapped to the template BSM_Router.

So what has been illustrated here? If you say that the resource that is being evaluated is a cdm:sys.UnitaryComputerSystem, then you have mapped it to two templates, BSM_Node and BSM_Router, so you end up with two instances of this resource in the SCR. One instance will be under Servers->All, the other instance will be under Network->Routers. In most cases this is not good, and this is where instance based template mapping comes in. If you go back and look in the CDM_TO_TBSM4x_MAP_Templates.xml document, you will see the document has associated cdm:sys.UnitaryComputerSystem with the template BSM_Node, but before this mapping is finalized, it will apply the policy "NotComputer".

The policy has one rule, and this rule checks the attribute "cdm:Type" for several values, one of which is "Router". Because the object in this example has "Router" for the value of "cdm:Type", we have a match and the template definitions at the bottom of the policy are applied, not the templates named for BSM_Node. Since the policy in this example does not have a template definition, the final outcome is that this resource is not assigned to BSM_Node and therefore is only instantiated as BSM_Router.

Functional Groups

TADDM Business applications group resources within a business application by functional groups. When editing a business application, the functional group to which one or more resources belongs to can be seen in the "Include" pane of the Create Business Application Components dialog box.

But when the topology for a business application is shown in TADDM, the functional groups are not shown.

TBSM on the other hand includes the functional groups when a TADDM Business Application is shown.

For every person that likes seeing the functional groups in TBSM, there is one that prefers not to see them. For those that prefer that the TBSM topology not include the functional groups, making a few changes in the XML configuration files can mask the functional groups.

The first change is to the CDM_TO_TBSM4x_MAP.xml file. TBSM cannot simply ignore the functional group, because the relationships associated with the functional group are needed to tie the business application to the underlying resources. Therefore, what you will do is create a composite, combining the business application and its functional groups. To do this you have to create a policy in CDM_TO_TBSM4x_MAP.xml and map cdm:app.Application, the CDM class representing a business application, to this policy.

You then need to map cdm:app.Application to the MergeFunctionalGroup policy.

Both the policy and the mapping need to go within the <CompositeDefinitions> tags. The specific location within these tags is not important, although to help with maintaining your customization, you might want to keep all changes together.

So to restate the changes to CDM_TO_TBSM4x_MAP.xml, the mapping statement says that for each resource of class cdm:app.Application, run policy MergeFunctionalGroup. The MergeFunctionalGroup policy follows any cdm:contains relationships that the resource may have, and if the target is either: cdm:app.FunctionalGroup, cdm:app.AppServerFunctionalGroup, or cdm:app.SoftwareModuleFunctionalGroup, then the resources are merged into a composite. The resulting behavior is that the application will inherit the relationships of the functional group, thus tying the application to all of the children of the functional group.
So thus far you have merged the application and functional group resources into one resource. One more change is needed. You have to remove the template mapping for the functional groups so that they are not also displayed as separate resources.

Edit CDM_TO_TBSM4x_MAP_Templates.xml and change the BSM_AppLogicalGrouping mapping sothat the functional group classes are commented out.

After changing the CDM_TO_TBSM4x_MAP.xml and CDM_TO_TBSM4x_MAP_Templates.xml files, stop and then restart the toolkit. After the toolkit has finished initializing, run the following command:

When the command finishes running, from the TBSM UI invalidate the Imported Business Service and then refresh the Service Navigation tree. You will see that the functional groups are no longer visible in the UI.

The above topology is a bit hard to read, but looking at the Service Navigation tree you can see that the functional groups are no longer in the tree.

Event to Resource Mapping

The previous sections have focused on discovery, importing data from TADDM, and populating the Service tree and SCR in TBSM. One challenge is to take events received by any of the data fetcher sources, such as NetCool/OMNIbus, and associating these events with the resources that have been imported from TADDM. These events can be from an endless number of sources: Tivoli Monitoring 6.1 or later versions, Tivoli Enterprise Console adapters (customer written, third-party written, IBM written), NetCool/OMNIbus probes, or the mainframe, and the contents of each of these events spans the spectrum from extremely rich to the bare minimum.

In order to associate an event with an imported TADDM resource, you must have some piece of data that is common in both the event and the data imported from TADDM. Every resource imported from TADDM will have one or more BSM_Identity fields built, this occurs as part of the import process. This field will be used to associate events with a resource. The EventIdentifierRules.xml file controls the content of a resource's BSM_Identity field. On the event side, the EIF Probe rules file is responsible for building a BSM_Identity field for the event.

The BSM_Identify is simply a convention, or default, that is used. The unique identifier could be another label specified with the field attribute on the rule definition.

The bottom line here is, there isn't a magic bullet that is going to associate any event from any event source with any and all resources discovered by TADDM. A little work will be needed to make this association. Out of the box, TBSM will associate Tivoli Monitoring 6.x events with Tivoli Monitoring resources that are in TADDM and z/OS events with z/OS DLA resources that are in TADDM.

The TBSM Administrator's Guide discusses the EventIdentifierRules.xml file and the syntax of the file. This should be reviewed to understand the basics of this file.

Defining a Resource's BSM_Identity

The EventIdentifierRules.xml provided with the product will build BSM_Identity values for imported TADDM resources for several specific cases, these cases include: Tivoli Monitoring 6.1 or later version event and resource mapping, z/OS DLA resource mapping, and generic generation of hostname and IP address identifiers. This file can be modified to fit the needs of the events received, and is not restricted to Tivoli Monitoring events or the cases discussed below.

IBM Tivoli Monitoring Resources

The EventIdentifierRules.xml file contains a policy that will build a BSM_Identity that will associate all Tivoli Monitoring 6.x events with their appropriate TADDM resources. If you are running a TADDM version earlier than 7.1, in-order for this to work, the TMS DLA must be run and the book imported into TADDM using the ldfxidml.sh script. Do NOT load the book with loadidml.sh. If you are running TADDM 7.1 or later version, please use loadidml.sh to load the book.

Tivoli Monitoring's unique identifier for resources is the managed system name. All Tivoli Monitoring 6.x events will contain this value and the resources in the TMS DLA book will have the managed system name as an attribute of each resource (cdm:ManagedSystemName). TADDM does not natively discover the managed system name if it discovers a resource that is being managed by Tivoli Monitoring 6.x. By loading the TMS DLA into TADDM, TADDM will reconcile the resources in the book with those resources previously discovered, associating the managed system name with the natively discovered resources. When TBSM imports the resources, it will receive the cdm:ManagedSystemName attribute, which contains the Tivoli Monitoring managed system name.

The EventIdentifierRules.xml file contains a policy and a mapping used to generate the BSM_Identify associated with an Tivoli Monitoring resource.

The mapping says that the policy "ITM" can be applied to any class and the policy says that if the resource contains an attribute named "cdm:ManagedSystemName", then a BSM_Identity will be built for this resource containing the value in the "cdm:ManagedSystemName" attribute.

An important piece, which will be discussed later in the paper, is that the NetCool/OMNIbus event will have a field in it called BSM_Identity. For Tivoli Monitoring events, this field will contain the managed system name.

The result of this is that any Tivoli Monitoring event received by NetCool/OMNIbus will be mapped to the associated TADDM imported resource because the imported TBSM resource will have a BSM_Identity containing the managed system name that matches the BSM_Identity field in the NetCool/OMNIbus event.

Hostname and IP Addresses

A second policy in EventIdentifierRules.xml will build BSM_Identity's containing the hostname and/or the IP address of any operating system resource. The thought here was that events that need to be posted against TBSM resources that represent an operating system contain either the hostname or the IP address of the computer system. So there is a likelihood that the hostname and IP address are valuable for mapping events for operating system resources.

The policy for building the BSM_Identity for this case is a bit more complex than the Tivoli Monitoring 6.1 policy, and it requires some familiarity with the CDM. Within the policy there are two rules: allIPAddresses and allIPFqdn.

Assume that the resource has the class "cdm:sys.windows.WindowsOperatingSystem", this satisfies the mapping, therefore the rule looks to see if this resource has a "runsOn" relationship with the OS as the source, if so it then checks to see if the target of the "runsOn" has a "contains" relationship with an "net.IpInterface", if that is true it then looks for a "bindsTo" relationship from the "net.IpInterface" with either an IpV4Address, an IpV6Address, or an IpAddress object. If all of this is true, it takes the value of the "cdm: DotNotation" attribute, assuming it does not begin with "10.", "192.", or "169.", and creates a BSM_Identity attribute for the resource. If your network uses IP addresses beginning with "10.", "192.", or "169.", possibly behind a firewall, then the rule must be changed accordingly.
Familiarity of the CDM is helpful here in understanding the relationships between the different CDM classes. To follow this example, it is best to open the CDM.htm that is shown in Section 6.1, and select the "Computer System" area of the graphic. This will open a schematic of the Computer System model. Then locate the OperatingSystem class and follow the links discussed below.

The hostname rule is similar to the IP address rule. Assume that the resource has the class "cdm:sys.windows.WindowsOperatingSystem", the rule looks to see if this resource has a "runsOn" relationship with the OS as the source, if so it then checks to see if the target of the "runsOn" has a "contains" relationship with a "net.IpInterface" object, if that is true it then looks for a "bindsTo" relationship from the "net.IpInterface" with either an IpV4Address, an IpV6Address, or an IpAddress object, and if that is true it looks to see if the source of the "bindsTo" is the target of an "assignedTo" relationship which has a "net.Fqdn" as a source. At this point the contents of the "cdm:Fqdn" attribute is used to create a BSM_Identity assuming the value is not "localhost". Most computers have a "localhost" binding, so this is ignored.

There are several mappings that specify this policy is associated with each of the CDM defined operating systems.

z/OS BSM_Identity

There are several policies in the EventIdentifierRules.xml file that are associated with the CDM classes used by the z/OS DLA, which are not discussed in detail here. The process for understanding these mappings is the same as what was described in the previous policies. The point to be made here is that the information in these mappings takes into account information that is available in events and messages generated by the event pump for z/OS, so the mapping of events in the future should be fairly straightforward.

Default BSM_Identity

If the EventIdentifierRules.xml file does not produce a BSM_Identity, then during the process of loading component registry objects into the TBSM service model, TBSM will assign an identity. The default BSM_Identity is comprised of the resource's name, TADDM GUID, and the CDM class.

This value is not expected to ever match any event received by NetCool/OMNIbus. You probably stand a better chance of visiting Mars than seeing the TADDM GUID in an event. The default BSM_Identity ensures that each resource has a BSM_Identity, and it allows you to easily associate the resource with TADDM's instance because it contains the GUID.

If you have an imported resource that only has a BSM_Identity with the preceding format, this means that none of the policies in EventIdentifierRules.xml matched this resource.

Defining an Event's BSM_Identity

This paper has just described how the BSM_Identity is built when a resource is imported; you now need to focus on how this field is built when an event is received by NetCool/OMNIbus. On the event side, the NetCool/OMNIbus EIF Probe rule file, tivoli_eif.rules, contains logic that will set the BSM_Identity for two cases:

- If the event contains an origin slot, then the origin is used.
- If the event contains a situation_name slot, this indicates that this event is from Tivoli Monitoring 6.1 and then the BSM_Identity is set to the situation_origin. The situation_origin is the Tivoli Monitoring managed system name.

The syntax of the NetCool/OMNIbus Probe Rule files is discussed in the *IBM Tivoli NetCool/OMNIbus Probe and Gateway Guide*, a link to this can be found in the references section below.

If changes are made to the EventIdentifierRules.xml file to build additional BSM_Identity fields, then corresponding updates need to be made to the Tivoli_eif.rules file so that the two match. This field can consist of a single value:

- @BSM_Identity = $situation_origin

or it can be constructed of a combination of values:

- @BSM_Identity = $situation_name + ":" + $situation_origin + ":" + $situation_displayitem + ":" + $ClassName

Which value or values are used to build the BSM_Identity is dependent on what data is available both within the event, and within the attributes of a resource.

From this common data, a field needs to be built that will uniquely identify the resource.

The Tivoli_eif.rules file is located in /opt/IBM/tivoli/netcool/omnibus/probes/<os>.

The import concept to understand here is that for events to be mapped to the correct TBSM resource, the value placed in the BSM_Identity field of the NetCool/OMNIbus events must match what the BSM_Identity value of the resource was set to during the TBSM import from TADDM.

Service Component Repository

As was mentioned in the introduction, all "physical" resources received from TADDM are placed in the Service Component Repository (SCR). The structure of the SCR is defined in the file BSM_Templates.radsh, and the

SCR is only visible in the Service Administrator mode.

While it is suggested that you do not change the BSM_Templates.radsh file, creating a file composed of RADShell statements can modify the structure of the SCR. Using the BSM_Templates.radsh file as an example, you can see the "Unix" portion of the tree is built with the following statements:

```
addServiceInstance(
new String\[\] \{"SCR\_TopLevelAggregateTemplate" \},
"SCR\_Servers\_Unix",
"Unix",
"",
"Standard",
null,
null,
null,
"REGULAR"
);
clearAllInstanceIDFieldValuePairs(
"SCR\_Servers\_Unix"
);
addUserPreferencesForInstance(
"SCR\_Servers\_Unix",
"Order",
""
);
addUserPreferencesForInstance(
"SCR\_Servers\_Unix",
"Longitude",
""
);
addUserPreferencesForInstance(
"SCR\_Servers\_Unix",
"Latitude",
""
);
addUserPreferencesForInstance(
"SCR\_Servers\_Unix",
"UseGIS",
"false"
);
addUserPreferencesForInstance(
"SCR\_Servers\_Unix",
"classnamefilter",
"'cdm:sys.sun.SunSPARCUnitaryComputerSystem','cdm:sys.sun.SunSPARCVirtualComputerSystem',
'cdm:sys.openvms.OpenVmsUnitaryComputerSystem','cdm:sys.linux.LinuxUnitaryComputerSystem',
'cdm:sys.ipso.IPSOUnitaryComputerSystem', 'cdm:sys.hpux.HpUnitaryComputerSystem',
'cdm:sys.aix.AixUnitaryComputerSystem'"
);
```

The "addServiceInstance" statement defines the "Unix" portion of the tree.

The "addUserPreferencesForInstance" statement defines the attribute "classnamefilter". This attribute is used to determine which CDM classes will be placed under the "Unix" node. The "classnamefilter" is visible in the additional attributes in the Service Viewer pane.

The contents of a node in the SCR can be altered by either modifying the "classnamefilter" in the Service Viewer or by using radshell to alter the "classnamefilter" attribute.

Rather than using a classnamefilter, some elements in the Component Registry tree are controlled by either a primarytemplatefilter or othertemplatefilter. Rather than using a CDM class name, this filter uses a template name. An example of this can be found in the definition of the SCR_SybaseDatabases definitions:

This restricts the contents of the Sybase database instances to those using the templates: BSM_SybsaeDatabaseServer or BSM_SybaseDatabase.

An "addServiceInstanceDependency" statement builds the structure of the SCR, for the "Unix" node the following statement indicates that "Unix" is a dependent of SCR_NodesRepository.

The SCR_NodesRepository is the "Servers" node, as shown by the following statement:

Runtime Monitoring

The TBSM Discovery Library toolkit runs as a service/daemon, and therefore does not have a user interface from which it can provide feedback. The state of the service can be monitored through the service's message log. TBSM provides three levels of messaging: Informational, Warning, and Error. Error provides minimal logging and is recommended for normal operation. Warning provides additional feedback when additional information is needed for longer operations as an assurance that some work truly is being done. Informational is used for debugging problems, and is not recommended because of the additional overhead.

TBSM also provides three levels of tracing: Min, Mid, and Max. Tracing is generally only used by IBM Software Support to debug problems. Min should be used for normal operation, and Max for problem determination. Max can be very I/O intensive and is only recommended if requested by IBM service personnel.

The message and trace logs are written to the $TBSM_HOME/XMLtoolkit/log directory. The message log is msgGTM_XT.log. The trace log is traceGTM_XT.log.

Error Message Logging

When TBSM begins a TADDM import, the following message will be written to the message log:

GTMCL5292I: Beginning CMDB import. Bulk: <t/f> Hostname: <taddmHostName>

Where:

- <t/f> is "true" if this is a bulk import and false if this is a delta import.
- <taddmHostName> is the TADDM Server that the import is running against.

When all data has been received from TADDM, the following message is displayed. The remaining processing is within the TBSM database.

GTMCL5283I: Data retrieval completed.

When the TADDM import has finished, one of three messages will be displayed:

- **GTMCL5306I: CMDB import completed successfully, no additional resource were found.**
- This message is only issued for a delta, and indicates that the delta finished successfully, but there were no new, changed, or deleted resources in TADDM, so no updates were made in TBSM.
- **GTMCL5293I: CMDB import completed successfully.**

- This message is issued for both a bulk and a delta import, and indicates that the import finished successfully, and that changes were made in the TBSM database.
- **GTMCL5294W: CMDB import did not complete successfully.**
- This message is issued for both a bulk and a delta import, and indicates that an error occurred during the import and the import did not finish successfully. The log should be examined for exception messages prior to this message.

Warning Logging

With respect to TADDM imports, the warning level provides additional feedback as the import runs. Some bulk imports can take an extended period of time, and the time between the GTM5292I and one of the three completion messages mentioned above can leave you wondering whether there really is work being done. When messaging is set to Warning, two additional messages are written to the log:

- **GTMCL5282I: Requested data from the database. Issued: <operation> Received <n> instances of the requested data.**
- This message includes the type of request made to TADDM and the number of instances of data that were returned. The type of request can be either a find() or a
findChanges(). The find() is used for bulk imports and requests all data for a particular class. The findChanges() is used for delta imports and requests any changed objects for a particular class within a given timeframe. This message will be written for each TADDM base class that is of interest to TBSM.

Informational Logging

When the message logging is set to Informational, the messages written to msgGTM_XT.log can be fairly verbose. Generally each step of the import process is recorded. Additional messages include:

- **GTMCL5321I: Issue generateExplicitRelationship. Delta Mode <true|false>.**
- **GTMCL5322I: The generateExplicitRelationship has finished**
- **GTMCL5317I: Request data from the CMDB database: <classname>**
- **GTMCL5318I: Begin SQL file execution: <sqlName>**
- **GTMCL5319I: SQL file execution complete.**
- **GTMCL5258I: Starting Transform: <transformName>**
- **GTMCL5259I: Transform complete: <transformName>**

Troubleshooting

Known Issues:

The most common issue is that a resource is not displayed in the TBSM UI. There are three general causes for this:

- The templates are not loaded. If this is the case, then run the following command:
- UNIX: cat $TBSM_HOME/install/BSM_Templates.radsh | $TBSM_HOME/bin/rad_radshell
- Windows: type $TBSM_HOME/install/BSM_Templates.radsh | $TBSM_HOME/bin/rad_radshell
- A template mapping was recently changed or is missing. If the CDM_TO_TBSM41x_MAP.xml or CDM_TO_TBSM41x_Templates.xml files were recently changed, then validate the change and run with the old copy if resources that were previously visible are no longer visible.

Verify that the CDM class representing the resource has a mapping in the CDM_TO_TBSM41x_Templates.xml file.

- The dependency counters in the database are incorrect. To correct this, run the "utils -e validate_dependency_counters.xml" command. After running this command, select the Imported Business Service in the UI, invalidate it, and then refresh the Service Navigation tree.

Data Collection

If problems occur, the first place to look is in the msgGTM_XT.log file. This file records all exceptions detected by the toolkit. If the msgGTM_XT.log file does not help to resolve the issue, and IBM service is called, the following files should be saved and sent in:

- The contents of the $TBSM_HOME/XMLtoolkit/sdk/log directory
- Run "utils -e collect_db_info.xml", which writes a file to the XMLtoolkit/log directory
- The contents of the $TBSM_HOME/XMLtoolkit/log directory
- The filter files located in $TBSM_HOME/XMLtoolkit/config/filters
- The customized XML files located in $TBSM_HOME/XMLtoolkit/xml:
- CDM_TO_TBSM4x_MAP.xml
- CDM_TO_TBSM4x_MAP_Templates.xml
- EventIdentifierRules.xml
- LabelingRules.xml
- The two or three files in $TBSM_HOME/XMLtoolkit/xml/scr/out associated with the bulk or delta import. Each bulk or delta import will generate two or three files. They have the format: taddm.<date>.<time>.refresh.xml.<tag>. The <date>.<time> corresponds to when the import was started. Samples are:
- taddm.2007-08-09T19-36-17Z.refresh.xml.1.sql
- taddm.2007-08-09T19-36-17Z.refresh.xml.2.label <may not be generated>
- taddm.2007-08-09T19-36-17Z.refresh.xml.2.sql

XML File Syntax Errors

If a problem occurs with an XML configuration file after editing the file, simple syntax errors in XML files can be found by attempting to open the XML file in a browser. This will not check whether the file follows a specific schema, but it will verify the basic XML syntax.

Analyzing the SQL Tables

The SCR is comprised of a series of tables and views stored in TBSM's PostgreSQL database. The schema is not published, but a series of SQL statements are provided that give insight as to the type and quantity of data that has been imported from TADDM. The "utils -e collect_db_info.xml" command will run these SQL statements and collect database specific information. An example of the information provided is:

- A list of the CDM classes imported and the number of each
- A list of CDM classes mapped to templates and to which template they are mapped.
- A list of CDM classes that are not mapped to a template.

- A list of CDM classes that are not mapped to a template and also not part of a composite object.
- Types of CDM relationships received and the number of each.

  The output from the utils command is written to the $TBSM_HOME/XMLtoolit/log directory.

  Depending on the size of the database, some of the SQL queries may take a long time to complete.

Loading DLA Books into TADDM

If problems occur after loading a DLA book into TADDM, the following files are required to diagnose the problem:

- The original DLA book XML file
- The TADDM bulk load results file. This file is located in $COLLATION_HOME/bulk/results. The results file will have the same file name as the originating XML file, but will have a results extension.
- The API server log. This file is: $COLLATION_HOME/log/services/ApiServer.log.
- The topology manager log file. This file is: $COLLATION_HOME /log/services/ToplologyManager.log.
- The bulk loader log file. This file is: $COLLATION_HOME /log/bulkload.log.

  In the $COLLATION_HOME /etc directory, the file bulkload.readme contains information on the bulk loader.

Removing All Resources from the SCR

If at some point the data from TADDM needs to be removed from the SCR, this is a fairly simple process. The first step is to remove all of the data from the tables used by the toolkit. To do this:

- Stop the TBSM toolkit
- Run the command, "setdbschema -U <dbuser> -f a"

  The setdbschema command will drop and recreate all of the tables associated with the toolkit, so all data previously imported from TADDM will be deleted.
  After running the setdbschema command, from the TBSM UI, invalidate the Imported Business Service.

  Invalidating will force the ESDA associated with the toolkit to run, because the toolkit's tables are now empty, this will force the deletion of any records associated with the toolkit. When the invalidation is finished, refresh the Service Navigation tree. The invalidate might take some time if there is a large amount of data under the Imported Business Service.

  An alternative to invalidating the Imported Business Service is to recycle the TBSM Data Sever. In TBSM 4.1, the process of restarting the TBSM Data Server forces the ESDA to run, but this is no longer true in TBSM 4.2.

Best Practices

This section specifies best practices for TBSM integration with TADDM. It sets a series of recommendations designed to promote more effective integration and a positive user experience. It includes suggestions to keep TBSM-imported TADDM data intact, improve the visual experience across products, utilize the functions provided by both products and more. Each recommendation is discussed in detail with scenario descriptions and solution options.

Impact of Change History Pruning on TBSM Delta Import from TADDM

When TBSM initiates a delta import with TADDM, it relies on the TADDM change history information to determine which objects have been created, modified or deleted in TADDM since the last import. Only the changed objects will be imported into TBSM.

If a database administrator prunes change history information from the TADDM database after the last TBSM import, but before the next delta import, it might affect the next import. Some data changed after the last import and before the change history pruning will not be imported into TBSM through the delta import, because the change history information has been pruned.

TADDM's guideline for database administrator is to prune the change_history_table data that is more than 4-6 weeks old. *See TADDM 7.1.2 Administrator's Guide, "Chapter 5. Polulating the database", Section "Maintenance tips for database"*. If your database administrator has been following this guidance and the TBSM administrator is using the default import interval, which is less than a day, TBSM delta import will not be affected by normal TADDM change_history_table pruning.
If TBSM has not imported data for more than 4 weeks, for example, TBSM has been down for more than 4 weeks or TBSM is being installed more than 4 weeks after TADDM was installed, the solution is to rebuild the TADDM relations table and then run a TBSM bulk import. This process varies depending on the level of TADDM:

- TADDM 7.1 and earlier
- On the TADDM server, run: "/opt/IBM/cmdb/dist/bin/explicitrel.sh 0" (that is a zero)
- On the TBSM server, run: "/opt/IBM/tivoli/tbsm/XMLtoolkit/bin/cmdbdiscovery.sh -b -r -e"
- TADDM 7.1.2 and later
- On the TBSM server, run: "/opt/IBM/tivoli/tbsm/XMLtoolkit/bin/cmdbdiscovery.sh -b -r -e"

Rebuilding the TADDM relations table can be a lengthy process, so plan accordingly.

How to orchestrate Discovery, Bulkload, TADDM Domain to eCMDB Synchronization, and TBSM Import from TADDM

The timing of TADDM discovery, bulkload, and synchronization operations can impact TBSM data imports. This section describes the interaction between these operations and the effect of these operations on TBSM imports of TADDM data.

TBSM data import from a stand-alone TADDM server

Figure 12 shows TBSM configured to import data from a typical TADDM stand-alone server in a non-enterprise environment. Resources may be discovered by the TADDM server, bulkloaded into the TADDM server via IDML books or manually added through TADDM UI or API. TBSM can import data from the TADDM server.
TBSM data import will check to see if a TADDM discovery or bulkload is running before starting the import. If one of these TADDM operations is in progress, TBSM will wait for it to complete before starting an import from TADDM. So a TBSM data import cannot start while a TADDM discovery or bulkload is in progress.

TBSM data import from an enterprise TADDM server

Note: TBSM does not support importing from TADDM 7.1 or earlier eCMDB server. While it can be made to work if care is taken with regard to timing between domain discoveries, eCMBD synchronization, and TBSM import, it is not recommended.

Figure 28 shows TBSM configured to import data from a typical TADDM enterprise **environment, consisting of a TADDM eCMDB (Enterprise Configuration Management Database) and multiple TADDM domains connected to that eCMDB.

Resources can be discovered, bulkloaded, or manually created through the TADDM UI or API at the TADDM domain server. Resources can also be bulkloaded or created through the TADDM UI or API at the eCMDB server. An eCMDB synchronizes information from the TADDM domains.

In the TADDM enterprise environment, TBSM integrates to TADDM by connecting to and import data from TADDM eCMDB server.

Similar to stand-alone TADDM server environments, TBSM data import checks to see if a bulkload is in progress on the TADDM eCMDB server before it starts the import action. (TADDM eCMDB server does not provide discovery function.) If a bulkload is in progress TBSM will wait until the bulkload completes before it starts the data import from TADDM. So a TADDM eCMDB bulkload and a TBSM data import from an eCMDB server cannot be performed in parallel.

A TBSM delta import of data from TADDM relies on change history data present in the TADDM database. In TBSM-TADDM enterprise integration environments, make sure change history data is synchronized from domains to the eCMDB. To enable a TADDM domain to synchronize change history data with an eCMDB, on the eCMDB computer:

- TADDM 7.1.2
- Make sure the following line **is not** listed in ** $COLLATION_HOME/etc/sync/table.skip:
- *change_history_table*
- TADDM 7.2
- Make sure the following line **is** listed in $COLLATION_HOME/etc/sync/table.extra:
- *change_history_table*

If TADDM is configured as suggested above, a TBSM delta import can be done at any time, irrespective of TADDM domain-eCMDB synchronization.
Because a TBSM bulk import from TADDM retrieves all CI types needed by TBSM from TADDM, it can be done at any time and does not require the configuration steps discussed above.

If a TADDM domain-eCMDB synchronization is in progress when a TBSM import is started, any updated CIs will be retrieved during the subsequent TBSM delta import. If a TBSM import from TADDM is initiated while a bulkload or discovery is running in TADDM, the TBSM import will wait until that operation completes.

Summary

The important points to remember about performing TBSM data imports from TADDM are the following:

Stand-alone TADDM server environments:

If a TADDM discovery or bulkload is in progress, TBSM will recognize that discovery or bulkload is in progress and wait until it is completed before continue with data import.

TADDM enterprise environments:

If a TADDM discovery or bulkload is in progress, TBSM will recognize that discovery or bulkload is in progress and wait until it is completed before continue with data import.

Make sure the following line is not listed in the eCMDB file $COLLATION_HOME/etc/sync/table.skip to enable TBSM delta data imports from TADDM:
change_history_table

How to Keep TADDM and TBSM Object Display Names Consistent

Components discovered in TADDM usually have generated display names in TADDM. The display names are imported into TBSM during data import, so the same display names are displayed in TBSM.

Both TADDM and TBSM allow you to change the display name of an object. If you want to change an object's display name and see the same name used in both TADDM and TBSM, the display name should be changed in TADDM. TBSM data import will load the changed display name into TBSM and use it as display name on the TBSM UI. If you change the display name in TBSM, TADDM cannot obtain and use the change.

How to change a display name in TADDM

The following instructions illustrate how to change a display name in TADDM.

1. Select an object from the tree view or topology view on the TADDM UI, right-click, and then select **Edit** from drop-down menu.

2. An Edit Component window will open as shown in the screen capture below. Depending on whether a label has been set for the object before, the label field may be populated with a value or it may be blank.

3. Edit the **Label** field in the Edit Component window as shown in he screen capture below and enter the display name for the object you want to see on the TADDM UI. Then click the "OK" button to save.

4. Reload the TADDM UI and go to the same object. You will see its display name has changed to the new value set in step 3, as shown in the screen captures below.

5. After the next TBSM data import to pick up the TADDM changes, you will see the new display name shown on TBSM UI as well, as shown in the screen captures below below.

Where to Define Business Services

TBSM and TADDM both model business services and provide function to allow users to define business services, so is it best to create your business services in TBSM or TADDM?

TADDM models business applications and business services. A business service can include one or more business applications. It can also include other business services. TADDM users can manually create business applications or business services through the UI or API. TADDM also provides application descriptors to automate the creation and maintenance of business applications through discovery. TBSM can import TADDM's business applications and business services. Business applications in TADDM appear as business services in TBSM.

TBSM models business services. TBSM users can define business services manually, automatically through autopop rules, or through an ESDA. Business services defined in TBSM can be bulk loaded into TADDM using a discovery library adaptor via the TADDM loadidml script. Business services defined in TBSM are displayed as business services in TADDM.

This leads to the question, which is the best approach? The answer to this question is not as simple as simply TBSM or TADDM because the answer involves not just these two products, but also the overall system architecture. A basic order of precedence is:

- If TADDM application descriptor files are being implemented and deployed, and TADDM will discover all components of your business system, then TADDM is the location to build your business services.
- If TADDM application descriptor files are being implemented and deployed, but TADDM will only discover pieces of your business system, build the business service in TADDM using the application descriptor files. This business application can then be imported into TBSM and added to the business service built in TBSM that contains the additional components of the business service.
- If the components of the business systems are discovered through a variety of means, one being TADDM, the process will be to manually construct the business services, or an automated approach in TBSM is being used,

then TBSM is the location to build your business services.

When to use TADDM

Use TADDM to define and maintain your business services and applications when:

1. Application descriptors can be used to define and maintain your business applications automatically in TADDM.
2. All the components of your business application or business service are stored in TADDM.

Using application descriptors

IBM application descriptors are application tags that TADDM uses to automatically associate components with business applications, eliminating manual creation and maintenance of business applications. When application descriptors are used to define business applications, TADDM can collect the information defined in these application descriptors and create and maintain business applications automatically for you through discovery. During a TBSM data import from TADDM, these TADDM business applications are loaded into TBSM as business services. Any changes to these business applications that are detected by discovery and stored in TADDM will be imported into TBSM.

For more information regarding application descriptors, see the "Application descriptors" chapter in the *IBM Tivoli Application Dependency Discovery User's Guide*.

Benefit:

TADDM automatically takes care of business application creation and maintenance. Changes that TADDM detects and stores will be loaded into TBSM during data import. The use of business application descriptors eliminates the manual process of creating business applications and keeps your business application information up-to-date.

Manually define business applications and services in TADDM

The TADDM UI provides functions to manually create business applications and business services. Users can easily select and include the components from a list of available resources stored in TADDM. See the following instructions regarding how to define business applications and services using the TADDM UI.

Instructions for defining business applications:

1. Select **Create Business Application** from the Edit menu .

2. Input general information in the pop-up window and click **Next>>** .

3. Define application components by selecting the component from available component list on the left side and then click the **Add>>** button to add them to a functional group. The selected components will be included in the list on the right. When you finish selecting all the included components, click **Next>>**

4. Define administrative information if applicable, and then click **Finish**.

Instructions for defining business services

1. Create a business service. Select **Create Business Service** from the Edit menu.

2. Input general information in the pop-up window, then click **Next>>** .

3. Define business service components by select the component from available component list on the left side, click **Add>>** button to add them to the included list on the right. When you finish selecting all the included

components, click **Next>>**.

4. Define administrative information if applicable, and then click **Finish**.

When to use TBSM

TBSM can load data from TADDM as well as a variety of other data sources. These other data sources include NetCool/OMNIbus events and external databases. If your business service includes resources from a data source other than TADDM or is a combination of data sources that may include TADDM, you can manually define the business service in TBSM.

In addition to manually creating business services in TBSM, an automated approach can also be used. An automated approach can include using auto-pop rules if the events received from NetCool/OMNIbus are rich enough, or an ESDA can be written if data is kept in an external database.

If events are used for service creation, but they are not rich enough to build a complete service hierarchy, the events can be enriched using information from an external database, and then auto-pop rules can be used to create the business services. The details of this approach will be left for another paper.
TBSM business services can be imported into TADDM via a bulkload of TBSM resources.

Instructions for manually defining business service

1. After logging into the TBSM UI, under the "All tasks" view on the left panel, expand **Tivoli Business Service Manager** and click **Service Administration**.

2. The "Service Administration" panel will show up in the UI to the right. Click the **Create New Service** icon.

3. A service editor will be displayed on the right panel. Enter the information for your business service, and then click on the disk icon on upper-left corner to save.

4. After the save completes, the newly created services will be displayed under services.

Appendix

TBSM Configuration with TADDM 7.1

If TBSM 4.2 is importing data from TADDM 7.1, the following additional configuration must be made.

TBSM Configuration

The following sections include information about TBSM configuration

API JAR Files

The TBSM Discovery Library toolkit uses TADDM's Java API to retrieve data from TADDM. This API requires that the API jar files used by TBSM be the same as those used by the TADDM server. Therefore, before starting the toolkit, these files must be copied from the TADDM server to the TBSM server.
The files can be found on the TADDM server in /opt/IBM/cmdb/dist/sdk/lib and replace the files on the TBSM server in $TBSM_HOME/XMLtoolkit/sdk/lib.
Even though TBSM only uses a subset of the jar files, all of the files should be copied. The files that are required can change between releases; therefore it is safer to copy all of the files.

TADDM fix packs can contain changes reflecting changes to the CDM. These types of changes can affect the objects received by the TADDM API. Therefore, these files should be recopied to the TBSM server anytime that a TADDM fix pack is installed. Not updating these files on the TBSM server will result in random exceptions when data is copied from TADDM to TBSM.

These exceptions are written to the $TBSM_HOME/XMLtoolkit/log/msgGTM_XT.log file.

System Clocks

Delta imports rely on timestamps to determine which TADDM classes have updated objects. The timestamp is based on the previous import. If the TADDM system clock is behind the TBSM clock, there can be a window that will allow an object change in TADDM to not be seen by TBSM. The TADDM system clock should either be synchronized with the TBSM server's system clock, or ahead of TBSM's clock. The window occurs only if the TADDM clock is behind TBSM's clock.

TADDM Database Access - Performance

TBSM 4.2 allows the option of directly querying the TADDM database via JDBC for specific high-volume transactions. This can reduce the data import times for large TADDM data models. These transactions are associated with retrieving relationship data. Two configuration steps are required:

- Run the taddmdirect script to enable the JDBC function. Sample invocation: "taddmdirect -enable -t DB2 -h ktaddm2.ibm.com -d cmdb -p 50000 -s db2inst1". Issue "taddmdirect -?" for complete usage information.
- Run the setxmlaccess script for the TADDM database userid and password. Sample invocation: "setxmlaccess -taddmdbid db2inst1 -taddmdbpw dbpw1234". Issue "setxmlaccess -?" for complete usage information.

Enhancements in the TADDM 7.1.2 API negate the need for this function when importing from levels later than TADDM 7.1, but if you are running TADDM 7.1, take advantage of the taddmdirect feature.
Configuring the taddmdirect feature is strongly recommended.

TADDM Configuration

The following sections include information about TADDM configuration.

Properties

The TADDM property generateExplicitRelationships controls whether TADDM generates additional relationships defined in the CDM and used by the TBSM import. If these relationships are not available to TBSM, TBSM will not be able to correctly recognize and display the objects received from TADDM.

Set the TADDM property generateExplicitRelationships=false and the TBSM property DL_TADDM_ExplicitRelationship=true (the default is true for TBSM).

API JAR Files

Again, simply a reminder that if a TADDM fix-pack is installed, then the jar files in .../dist/sdk/lib must be updated on the TBSM server. It cannot be stressed enough about how important this is.

System Clock

Again, simply a reminder to avoid having the TADDM server's system clock fall behind the TBSM server's system clock.

Scheduling Discoveries

The TADDM 7.1 API will not prevent a discovery from starting during critical sections of the TBSM import, to prevent data synchronization issues, a TADDM discovery should not be started while a TBSM import is running.

w3-connections.ibm.com

Performance and Tuning (tbsm) - Tivoli Netcool/Impact Wiki

71-90 minutes

This section of the wiki includes performance and tuning information for Tivoli Business Service Manager. The following topics are available:

- BSM 4.2.1 - 6.1 Performance & Tuning Recommendations

Additional Server Tuning Suggestions

Service Navigation Threshold (Data Server and Dashboard Server)

In TBSM 4.2.1, the tree used in the Service Administration page uses the older Service Navigation portlet. From a coding point of view, this is different to the tree used in Service Availability page (Service Tree portlet), and is not reported in the queued statistics by ShowTreeListeners.jsp, as they are for the new scorecard only.

APAR IV09053, shipped with 4.2.1-TIV-BSM-FP003 for TBSM 4.2.1, uses the value set for the impact.sla.servicetree.maxupdates property to limit the number of updates that can be queued for these trees. impact.sla.servicetree.maxupdates is stored in RAD_sla.props, at the following location:

$TIP_HOME/systemApps/isclite.ear/sla.war/etc/rad/RAD_sla.props (Dashboard Server)

$TIP_HOME/tip/profiles/TBSMProfile/installedApps/TBSMCell/tbsm.ear/sla.war/etc/rad/RAD_sla.props (Data Server)

The value is commented out by default (i.e. no limit) with a setting of 1000. If memory use for the ServiceTreeUpdatesCommand$ServiceContainer class is growing, then consider commenting out this property and setting to an appropriate value. This will depends on the refresh rate set for the tree and the number of updates to the tree in a refresh period. If the limit is hit, updates are not lost. Instead the tree is refreshed.

A server restart is required for the change in value to take effect.

Port Timeout (Operating System)

The default Operating System port timeout can cause a lack of available TCP/IP ports. This can occur on busy servers which attempt to service many connections in a short amount of time. The result can cause the TBSM Dashboard Server to run out of memory with a large number of dashboard updates outstanding. The Heap dump will show a memory leak using one of the Linked Lists in RADClientUpdateHandler.java. To check for this problem on a system, before an Out Of Memory situation occurs, use 'netstat -a' and look for a high number of connections in a CLOSE_WAIT state.

To improve the situation significantly (on Windows), apply two Windows Registry modification. Add these two new DWORD values to the Windows Registry in the following location, on both the Dashboard and Data servers.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentConrolSet\Services\Tcpip\Parameters\

MaxUserPort REG_DWORD 0x0000fffe (65534)

TcpTimedWaitDelay REG_DWORD 0x0000001e (30)

Reboot boot systems to have the changes take effect.

On RHEL5 zLinux systems the default timeout value is 60 seconds. This value can be changed to 30 seconds by issuing the following command:

echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout

On other Operating systems, the defaults for TCP Timeouts are:

AIX: 75 seconds

Solaris: 180 Seconds

To view the current setting use:

# /usr/sbin/no -o tcp_keepinit

The output should be something like:

tcp_keepinit = 150

The above change will only take effect after a data server restart.

RMI Logging

If RMI logging is turned on for servers, it can significantly increase the amount of server I/O. The change below will lower the amount of information being logged in the trace file, diminishing I/O usage.

On Dashboard server file: $TIP_HOME/systemApps/isclite.ear/sla.war/etc/rad/RAD_server.props

Property to change : impact.server.rmilogging

Default Value : true

New Value : false

The above change will only take effect after a Dashboard Server restart.

On Data Server file: $TIP_HOME/profiles/TBSMProfile/installedApps/TBSMCell/tbsm.ear/sla.war/etc/rad /RAD_server.props

Property to change : impact.server.rmilogging

Default Value : true

New Value : false

The above change will only take effect after a data server restart.

Disable SLAM events

When the value of a rule or key performance indicator changes, TBSM sends a status (Class 12000) event for every service instance and rule combination. These events show up as purple status events and can be seen from the Rules tab in Service Details portlet. The generation and processing of these events, adds to the server's work load. If these events are not required, then disable them using the information in the following link:

ulimit

On Unix systems, the TBSM Servers will crash if the open file ulimit is set too low. In the trace logs there will be an exception indicating Too many open files. This exception indicates that the TBSM server process has exceeded a system-imposed limit on the number of files that can be open by a given process or user. You can increase the system limit for files.

The system administrator must add the following lines to the /etc/security/limits.conf file:

tbsmadm soft nofile 8192

tbsmadm hard nofile 8192

Where tbsmadm is the user ID that runs the TBSM server process. If your user ID is different, then replace the userid in the above lines as appropriate. The tbsmadm user may need to log out and log in order for the above change to take place.

To verify the change, run the following command from the shell prompt where the TBSM server will be started:

ulimit -a

The value reported for open files (-n) should be 8192.

Note: Make sure that 8192 or whatever value is chosen for the TBSM server process does not exceed the system-wide limit for the maximum number of open files. The system-wide limit resides in the /proc/sys/fs/file-max directory. The number chosen for the TBSM server should only be a fraction of the limit specified in file-max.

The system administrator needs to increase the values for the following kernel configuration tunables:

maxfiles - Initial (soft) maximum number of file descriptors per process (increase at least to 8192) maxfiles_lim - Hard maximum number of file descriptors per process (increase at least to 8192)

You can modify these parameters using either the System Administration Manager (SAM) utility or System Management Homepage (SMH) utility. SMH replaces SAM as the system management interface as of HP 11i v3 (11.31).

To enable the maxfiles changes, you must reboot the system.

The system administrator needs to add or increase the following values in the /etc/system file:

set rlim_fd_max - Hard maximum number of file descriptors per process (increase at least to 8192) set rlim_fd_cur - Initial (soft) maximum number of file descriptors per process (increase at least to 8192)

To enable the changes, you must reboot the system.

The system administrator needs to permanently change the ulimit setting of the user running the TBSM server. The ulimit setting specifies the number of open file descriptors for a given user. The change this setting:

1. Log in as the root user on the TBSM host.

2. To change the number of open files limit to 8192, enter the command:

chuser "nofiles=8192" username

3. Log in as username and check the new setting with the command:

ulimit -a

The command output should match the number you specified in step 2.

nofiles(descriptors) 8192

Disable automatic database start up

If you continue to receive the too-many-open-files exception after you change your system settings, you can disable the automatic start up of the TBSM database. By default, the database starts whenever you start the TBSM server. To disable the automatic start up of the TBSM database, complete the following steps:

1. Change to the directory $TBSM_DATA_SERVER_HOME/etc/rad.

2. In a text editor, open the file: RAD_sla.props.

3. Add the following property:

impact.sla.autostartdatabase=false

When you set impact.sla.autostartdatabase to false, manually start the TBSM database before your start the TBSM server. When you set this property to false, TBSM does not periodically check that the database is running.

To start the database, run the following command:

$TBSM_HOME/bin/rad_db start

LDAP Performance

Using TBSM, connected to LDAP with a large number of users and groups can affect TBSM Performance.

For example, when a customer creates or saves an already created service instance from the TBSM GUI, the console may hang for some time. To improve TBSM performance in this scenario, the following should be done

1. Limit Search base.

The Search base can be limited by introducing filters to restrict the amount of data returned.

Examples of limiting the search base for group and user:

<config:ldapEntityTypes name="**Group**" searchFilter="**(ObjectCategory=Group)**">

<config:objectClasses>**group**</config:objectClasses>

<config:searchBases>**OU=Monitoring,OU=Global,DC=corp,DC=ABC,DC=com**</config:searchBases>

</config:ldapEntityTypes>

<config:ldapEntityTypes name="**PersonAccount**" searchFilter="**(objectCategory=person)**">

<config:objectClasses>**user**</config:objectClasses>

<config:searchBases>**OU=Regions,DC=corp,DC=ABC,DC=com**</config:searchBases>

</config:ldapEntityTypes>

2. Enable dynamic cache, by setting enable to "true" for the **server.xml** files of the Data and Dashboard servers. For example, replace:

<services xmi:type="applicationserver:DynamicCache" xmi:id="DynamicCache_1254907162252" enable="false">

with

<services xmi:type="applicationserver:DynamicCache" xmi:id="DynamicCache_1254907162252" enable="true">

Note: Changes to server.xml require a server restart.

3. Increase search Limit.

This can be achieved by editing the **wimconfig.xml** file, see below:

<config:configurationProvider maxPagingResults="500" maxSearchResults="4500" maxTotalPagingResults="1000" pagedCacheTimeOut="900" pagingEntityObject="true" searchTimeOut="600000">

4. Reduce Search timeout.

This is also achieved by editing the wimconfig.xml file and changing the value for searchTimeOut. (see above)

Note: Changes to wimconfig.xml require a server restart.

Data Fetcher Considerations

Data fetcher processing is sequential in TBSM. The data for one Data Fetcher is processed before another is commenced. If Data Fetcher performance is an issue, a full examination of how much data the data fetcher is returning and the frequency of the data fetcher runs should be performed. Reducing the amount of data returned should improve overall performance.

If the data from data fetchers is just displaying numbers on the scorecard and not being used in status calculations, then it is not recommended to configure Data Fetchers which pull back large amounts of data. For example, in such a scenario, with caching disabled, 11,000 rows can take an average of 20 minutes to display KPIs on the scorecard.

The first step to speed up processing is to reduce the number of rows returned. Consider making a query with filters and only pull back the needed information, and no more. Make sure they have enough memory to hold this information, so maximum heap adjustments should be made accordingly.

Secondly enable caching by editing the data fetcher props file as follows:

impact.<nameOfFetcher>.enablecaching=true

impact.<nameOfFetcher>.maxnumberofsavedrows=<estimatedNumberOfRows>

If the Data Fetcher is fetching 5100 rows, set the maxnumberofsavedrows parameter to 5000.

The first fetch will take some time processing, but the subsequent fetches would only take seconds compared to the minutes of the first fetch. This change is only recommended for customers that pull back large amounts of data using data fetchers. TBSM considers > 5000 rows to be a large amount as it takes a large amount of time to process all these events in the state model after pulling the data from the database.

If performance is still slow and a very large number of rows is required then direct fetching should be used instead of Data Fetchers.

Client-side Java Virtual Machine Tuning

Within the client Web browser that hosts the TBSM Web Console is a JVM plug-in needed for running client side Java code. Just like a JVM running on either of the TBSM servers, the browser plug-in JVM can also be tuned to specify initial and maximum heap sizes. Typically, an untuned JVM plug-in may typically have an initial heap size of 4 MB and a maximum heap size of 64 MB, though these numbers may vary based on the platform used and amount of physical RAM on the system.

The graphical Service Viewer and the Active Event List (AEL) are the functions most affected by changes to JVM plug-in parameters. It might be possible to improve the performance of these applets by increasing the initial heap size (-Xms) to 64 MB and the maximum heap size (-Xmx) to 128 MB. Whether this configuration change is really needed or not depends on the size of the business service model displayed in the Service Viewer, or the amount of records displayed in the AEL.

The procedure to change the JVM plug-in tuning parameters can be different depending on the provider of the plug-in (for example IBM or Sun) and also depending on the system (Windows or UNIX). As an example, the following procedure illustrates how to access and set the JVM plug-in parameters for the IBM-provided 1.6.0 plug-in on a Windows system:

1. Open Control Panel -> Java Plug-in.

2. Click the Advanced tab.

3. In the text box under Java Runtime Parameters, type the following value:

-Xms64m -Xmx128m

4. Click the Apply button and then close the window.

After these changes are made, it might be necessary for you to log out and log back in to the TBSM console. For a complete list of the supported combinations of Java plug-ins, Web browsers, and platforms, see the TBSM Installation Guide.

Important: To change the JVM plug-in parameters on a supported UNIX system, navigate to the bin directory under the file system location to which the plug-in was installed and look for a shell script named either ControlPanel or JavaPluginControlPanel, depending on your Java version. Run this shell script to launch a GUI that looks similar to the equivalent interface on the Windows system.

Dashboard Server - Additional Suggestions

Canvas update interval multiplier

The update interval multiplier helps to control the frequency of automatic canvas refresh requests initiated by the Service Viewer for canvases containing business service models of different sizes. The default multiplier is 30.

For example, computing a larger, 100 item canvas takes longer than loading a smaller, 50 item canvas. Because of this, the refresh intervals of the larger canvas should be spaced apart so that the canvas is not constantly in a refresh state. The TBSM Web Console accomplishes this by computing a dynamic refresh interval by taking the amount of time spent performing the previous load request and multiplying it by the update interval multiplier constant. So, if the large service model in this example takes 2 seconds to load, a refresh of the same model is not attempted for another 2.5 minutes (5 x 30 or 150 seconds).

When considering a change to this parameter, keep in mind that there are lower and upper boundaries of 30 seconds and 180 seconds for the refresh interval. As a result, the update interval multiplier is useful only to a certain point.

Nonetheless, for **TBSM 4.2.x**, you can easily update the interval multiplier parameter on the TBSM Data Server by editing:

$TIP_HOME/systemApps/isclite.ear/sla.war/av/canvasviewer_simple.html

For **TBSM 6.1**, edit the following file instead:

$TIPPROFILE/installedApps/TIPCell/isc.ear/sla.war/av/canvasviewer_simple.html

Change the value for all occurrences of the UpdateIntervalMultiplier property to the new value. Because this is a client side property, you do not have to reboot the server for this value to take effect. However, you might need to log out and log on to the TBSM console.

Session Timeout

Session timeout represents the length of time a session can be idle, before it is invalidated. The setting is a value in minutes, with a default value of 30. While a session is valid, TBSM will continue to queue updates to scorecards etc. for the session. Because sessions will remain open unless logged out, TBSM recommends a maximum session timeout of 10080 - which is equivalent to 7 days (7*24*60 = 10080). The value for session expiry is stored in the **invalidationTimeout** setting, the following file:

$TIPProfile/config/cells/TIPCell/applications/isclite.ear/deployments/isclite/deployment.xml

Note: If clients become disconnected, without logging out, their session remain active until the invalidationTimeout period is met. This is also true if a user closes the application with logging out. If a user does not log out of the console themselves, an administrator can check which users are active and log them out - using Users and Groups -> Administrative User Roles portlet.

Note: The above functionality is only valid if the user IDs are individually defined. In cases where the user gets their role information via the Group, the user would not show up on the list at all.

Note: The Session Timeout valid is activity based, i.e. it only takes affect when the session is idle.

Scorecard Threshold

The invalidationTimeout parameter above is a safety net that keeps invalid sessions from staying in the system forever. The impact.sla.scorecard.updateListThreshold is another safety net in the TBSM code - which restricts the number of queued updates which TBSM will queue per scorecard.

However, by default, this is disabled (with a very high value set internally).

To update this value **for TBSM 4.2.x**, edit:

$TIP_HOME/systemApps/isclite.ear/sla.war/etc/rad/RAD_sla.props

on each Dashboard server and append the following parameter to the bottom of the file:

impact.sla.scorecard.updateListThreshold=xxx

Where xxx is set to a reasonable number, based on how many pending updates is a single "Scorecard cycle update" is going to generate.

For **TBSM 6.1**, edit:

$TIPPROFILE/installedApps/TIPCell/isc.ear/sla.war/etc/RAD_sla.props instead.

This will allow TBSM code to detect when a user is no longer retrieving updates and will perform its own clean-up (which could force the user to close and re-open the page containing the scorecard, if still logged on). If not logged on, a complete clean-up is done. This parameter only addresses issues related to service tree portlets, i.e. Service Availability page. It does not control updates for the Service Navigation portlet - used in Service Administration (see section below).

To determine how many pending updates is a single "Scorecard cycle update" is going to generate, use the following test, which works best if the scorecard does not have the "Service Visibility" option turned on:

1. Logon to TBSM and access the page with the scorecard

2. Open another browser session to monitor the ShowTreeListeners (see section below)

3. Go back to the session with the active scorecard and without closing it, access the "Users and Groups" > Manage Users page and then click on the "Create" button.

Note: Having the "WIM User Management" page open (with "Create a User" function) will stop the delivery of the pending updates to the scorecard

4. Go back to the browser session that is monitoring the ShowTreeListners. You'll notice that depending on the

scorecard refresh interval, the pending update counts will increase with every interval. Take note of how often the counts increase and by how much (you'll need to perform a refresh of the ShowTreeListeners browser page to get it to show the pending updates changes.

5. When testing are recording has been completed:

- cancel the "Create a User"

- return to the page containing the scorecard. Now, if you go back to the "ShowTreeListeners" browser session, you should see the accumulated pending updates begin to decrease and eventually all go away (for the one scorecard that accumulated the pending updates during the test).

The above information can be used to set a proper value for the threshold parameter. Take the average number of pending updates and multiply that by x number of cycles an Administrator would spend away from the scorecard, and still be expected to return back to the scorecard and have it accessible (i.e. where the page recovers and retrieves all pending updates. The length of a cycle will depend on the refresh rate of the tree. Please note that there are two reason we need to put a reasonable number here.

1. Not too small, where there accumulation of pending updates per cycle is higher than the threshold. A low value in this case would constantly give you an error on the page that say you need to re-open the page.

2. You do not want to set it too high, where you might have users that opened scorecards, abandon their sessions.

As those abandoned sessions will continue to accumulate pending updates, until

- the pending updates threshold is reached (and cleanup is initiated)

- the session invalidation period is reached (and cleanup is initiated)

- the number of pending updates consumes all available memory.

3. Allow TBSM Administrator to return to the scorecard page (after some admin action) to allow the pending updates to be received (without having to re-open the scorecard page).

ShowTreeListeners

This is a useful tool provided with TBSM 6.1 that allows one to dynamically monitor the Scorecard and see if there are any pending updates.

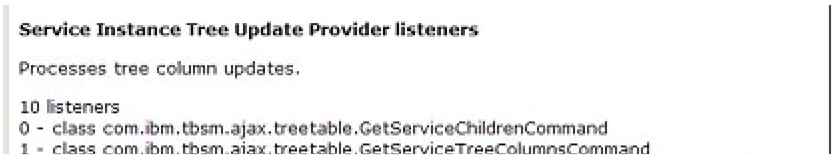
The following is the procedure that can be used to access the debug tool ShowTreeListeners.jsp (**Note**: No re-start of the Dashboard is required)

1. File location:

/$TIPProfile/installedApps/TIPCell/isc.ear/sla.war/rad/debug/ShowTreeListeners.jsp

2. Logon to the Dashboard and then Access the debug code, for example with the following URL https://yourhost:16316/ibm/sla/rad/debug/ShowTreeListeners.jsp

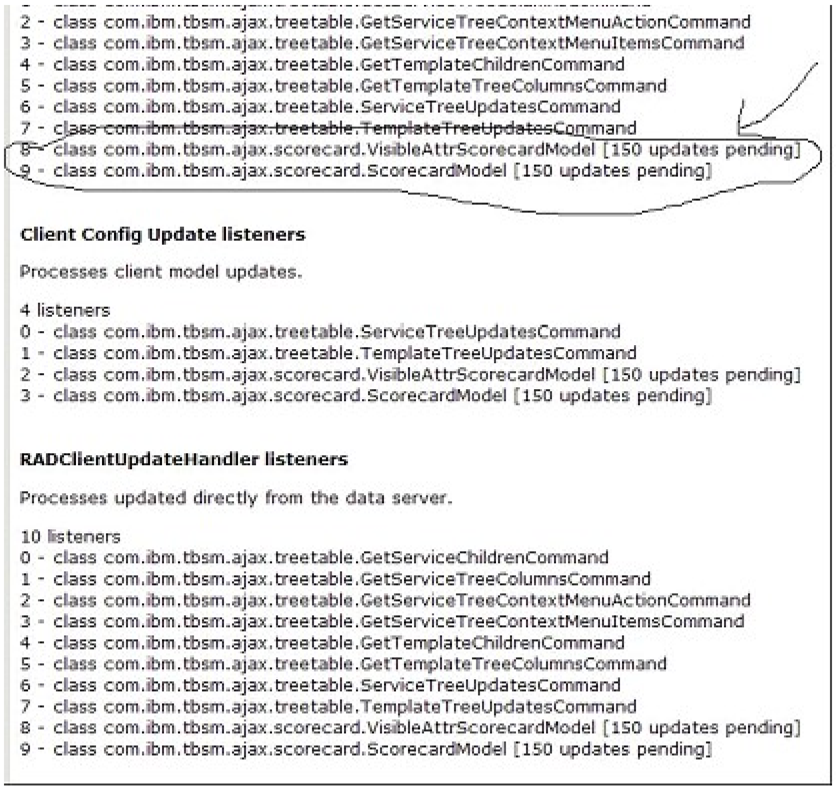
4. The jsp will show all registered listeners of the TBSM update services (including the ScorecardModel) and list the pending updates. If there is a model with a large number of pending updates that continues to grow, it may indicate a leaked model. If all consoles have been logged out, then there should no longer be any ScorecardModels shown.



Service Instance Tree Update Provider listeners

Processes tree column updates.

```
10 listeners
0 - class com.ibm.tbsm.ajax.treetable.GetServiceChildrenCommand
1 - class com.ibm.tbsm.ajax.treetable.GetServiceTreeColumnsCommand
```



```
2 - class com.ibm.tbsm.ajax.treetable.GetServiceTreeContextMenuActionCommand
3 - class com.ibm.tbsm.ajax.treetable.GetServiceTreeContextMenuItemsCommand
4 - class com.ibm.tbsm.ajax.treetable.GetTemplateChildrenCommand
5 - class com.ibm.tbsm.ajax.treetable.GetTemplateTreeColumnsCommand
6 - class com.ibm.tbsm.ajax.treetable.ServiceTreeUpdatesCommand
7 - class com.ibm.tbsm.ajax.treetable.TemplateTreeUpdatesCommand
8 - class com.ibm.tbsm.ajax.scorecard.VisibleAttrScorecardModel [150 updates pending]
9 - class com.ibm.tbsm.ajax.scorecard.ScorecardModel [150 updates pending]
```

Client Config Update listeners

Processes client model updates.

```
4 listeners
0 - class com.ibm.tbsm.ajax.treetable.ServiceTreeUpdatesCommand
1 - class com.ibm.tbsm.ajax.treetable.TemplateTreeUpdatesCommand
2 - class com.ibm.tbsm.ajax.scorecard.VisibleAttrScorecardModel [150 updates pending]
3 - class com.ibm.tbsm.ajax.scorecard.ScorecardModel [150 updates pending]
```

RADClientUpdateHandler listeners

Processes updated directly from the data server.

```
10 listeners
0 - class com.ibm.tbsm.ajax.treetable.GetServiceChildrenCommand
1 - class com.ibm.tbsm.ajax.treetable.GetServiceTreeColumnsCommand
2 - class com.ibm.tbsm.ajax.treetable.GetServiceTreeContextMenuActionCommand
3 - class com.ibm.tbsm.ajax.treetable.GetServiceTreeContextMenuItemsCommand
4 - class com.ibm.tbsm.ajax.treetable.GetTemplateChildrenCommand
5 - class com.ibm.tbsm.ajax.treetable.GetTemplateTreeColumnsCommand
6 - class com.ibm.tbsm.ajax.treetable.ServiceTreeUpdatesCommand
7 - class com.ibm.tbsm.ajax.treetable.TemplateTreeUpdatesCommand
8 - class com.ibm.tbsm.ajax.scorecard.VisibleAttrScorecardModel [150 updates pending]
9 - class com.ibm.tbsm.ajax.scorecard.ScorecardModel [150 updates pending]
```

Note: In the list above, we have two different flavors of scorecard. The first, is the standard scorecard. The second is from a scorecard that has the "Service Visibility" option turned on. These are two different scorecard instance sessions. When viewing a scorecard, a user would have one or the other on their portlet.

1 - class com.ibm.tbsm.ajax.scorecard.ScorecardModel [0 updates pending]

2 - class com.ibm.tbsm.ajax.scorecard.VisibleAttrScorecardModel [0 updates pending]

LTPA

The LTPA token is used for secure communications between the client and server. It is a cookie that contains the user domain and the user identifier of the caller, and is both encrypted and signed. For additional security, it has an absolute validity period which invalidates its usage after a defined period of time. The value for LTPA token expiry is stored in the **timeout** setting, the following file:

<TIP_INSTALL_DIR>/profiles/TIPProfile/config/cells/TIPCell/security.xml
The value for **timeout** is set in minutes with a default value of 1440, equivalent to 24 hours.

http Timeout Settings

To prevent premature I/O & KeepAlive timeout of http requests, especially when Web Server is remote and connection between the client & Web Server is slow, adjust the HTTP transport parameters of WebSphere Web Container

On Dashboard server file: $TIP_HOME/profiles/TIPProfile/config/cells/TIPCell/nodes/TIPNode/servers/server1/server.xml

Property to change : readTimeOut and writeTimeOut in parameters with "HTTP_3"

Default Value : 60

New Value : 180

For example:

<transportChannels xmi:type=""channelservice.channels:HTTPInboundChannel"" xmi:id=""HTTPInboundChannel_1255539271376"" name=""HTTP_3"" discriminationWeight=""10"" maximumPersistentRequests=""100"" keepAlive=""true"" readTimeout=""180"" writeTimeout=""180"" persistentTimeout=""30"" enableLogging=""false""/>

Data Server - Additional Suggestions

Following are some additional tuning suggestions for the TBSM Data Server to further reduce processing overhead. Review the following areas and consider implementing them based on the needs of your unique environment.

Service tree refresh interval

Changing the automatic Service tree refresh interval might help reduce server side workload (on the TBSM Dashboard Server) related to TBSM Web Console activity. The service tree refresh interval is set to 60 seconds by default.

The service tree refresh interval controls how frequently the TBSM Web Console requests an automatic service tree update from the TBSM Dashboard Server. If every client connected to the TBSM Dashboard is updated every 60 seconds, this might affect the Dashboard Server when there are a large number of concurrent consoles. To help mitigate this, you can increase the interval between refreshes.

For **TBSM 4.2.x**, edit the RAD_sla.props file in the $TBSM_DATA_SERVER_HOME/etc/rad/ directory:

#Service Tree refresh interval; in seconds - default is 60

impact.sla.servicetree.refreshinterval=**120**

For **TBSM 6.1**, edit the TBSM_sla.props file in the $TBSM_DATA_SERVER_HOME/etc/ directory.

Increase number of Events per Query (TBSM 4.2.1)

The impact.eventprocessor.numeventsperquery determines the size of the Event Processor Batch. In order to allow the event processor to recover a little quicker from situations where there is an event storm, set the value for this property to 100.

This change is required on the Data Server only. The property is stored in the following file:

$TIP_HOME/profiles/TBSMProfile/installedApps/TBSMCell/tbsm.ear/sla.war/etc/rad/RAD_eventprocessor.props

Property to change : impact.eventprocessor.numeventsperquery

Default Value : 10

New Value : 100

Turn off the TBSM Agent

If the TBSM Common Agent for ITM is not operational on the TBSM Server, then the component within the TBSM Data Server, which communicates with the Agent, can be disabled.

For the **TBSM 4.2.1** TBSM Data Server, edit the following file: $TBSMProfile/installedApps/TBSMCell/tbsm.ear /sla.war/etc/rad/RAD_agentservice.props

For **TBSM 6.1**, edit the following file instead:

/opt/IBM/tivoli/tbsm/etc/TBSM_agentservice.props

Property to change : impact.agentservice.autostartup

Default Value : true

New Value : false

The above change will only take effect after a data server restart.

If the TBSM agent is running and verbose logging is not required, ensure verbose logging is not turned on.

On Data server file: $TIP_HOME/profiles/TBSMProfile/installedApps/TBSMCell/tbsm.ear/sla.war/etc/rad
/RAD_agentservice.props

Property to change : impact.agentservice.verbose

Default Value : false

Recommended Value : false

Time Window Analyzer (TWA) Processing

A default TBSM installation will have TWA data collections enabled.

For **TBSM 4.2.x**, the .props file that controls TWA processing can be found here :

%TBSM_DATA_SERVER_HOME%\etc\rad or ${TBSM_DATA_SERVER_HOME}/etc/rad

impact.collectmetrichistory turns TWA data collection by TBSM on/off. If TWA collection is not required, this can be turned off.

Property to change : impact.collectmetrichistory

Default Value : true

Recommended Value : false (if TWA is collection not required)

For further information, consult this link:

If TWA is enabled, there are two additional TBSM 4.2.1 tuning recommendations (not required for TBSM 6.1 using DB2), which are beneficial in improving response times when testing the Time Window Analyzer (TWA) function. These two settings are both associated with Postgres, and therefore will require modification of the **%TBSM_HOME%\pgsql8\data\postgresql.conf** file on the Data Server. It is recommended to set both the **shared_buffers** and **effective_cache_size** parameters to a minimum value of 50,000, as follows, if the current value of either parameter is lower than 50000.

shared_buffers = 50000

effective_cache_size = 50000

Once the modifications have been saved to the postgres.conf file, stop the TBSM Data Server followed by the Tivoli Postgres Database. Then, restart the Tivoli Postgres Database followed by a restart of the TBSM Data Server.

For **TBSM 6.1**, the metric history collection thread can be shut off via a property in file /opt/IBM/tivoli/tbsm/etc
/TBSM_tbsm_metric_history_user.props:

# Switch to turn on metric history collection. The default is true.

impact.collectmetrichistory=false

Once again, the above change will only take effect after a TBSM Data Server restart.

Final Thoughts on TBSM Performance

To review, TBSM is primarily processor dependent (the number and speed of processors are two of the key factors); as long as sufficient JVM memory is configured (use the IBM PMAT tool to assist you in tuning TBSM for your own workloads and environments). You must be aware of the minimum and recommended hardware specifications for an optimal user experience. The TBSM minimum and recommended hardware tables are supplied in the Hardware for production environments section of this document for easy access and review.

Prior to beginning any in-depth performance tuning for TBSM, review the trace_*.log files that are created by both the Data and Dashboard servers. These logs are typically in the profiles /logs/server1 directory. Review any exceptions or error conditions to remove a functional issue from hampering overall application performance.

After functional processing is observed, two of the primary tuning "knobs" for TBSM are the "maximumHeapSize" and " mimimumHeapSize " values that control the memory allocation for each of the TBSM JVMs.

For review:

-Xms256m (minimumHeapSize) // Sets the initial memory to 256 MB (default)
-Xmx512m (maximumHeapSize) // Sets the maximum memory size to 512 MB (default; default value on TBSM 6.1 Data Server is 1200 MB)

After the upper memory setting for maximumHeapSize is established (through PMAT analysis), a good rule of Java tuning is typically to set the initial memory allocation to the same as that of the maximum size.

Again, one "size" does not fit all environments, so you might want to try setting the initial value smaller (or larger) and rerun the scenarios. Note that you should not set the minimumHeapSize value larger than the maximumHeapSize value, or the JVM will most likely not start.

After the Data and Dashboard servers are properly tuned, if Web Consoles using the Service Viewer feel "slow," review the Client side Java Virtual Machine tuning section on tuning the JRE plug-in JVM, and restarting the Console.

Every TBSM Environment is unique; with regard to tuning, ***one tuning "size" does not fit all***. What this means is that multiple factors come into play, such as number and speed of processors, available RAM, service model size, number of concurrent Web consoles, event rates, and the number of KPIs, to name but a few. JVM analysis is the correct way to ensure proper performance tuning is in place.

To do this, a regular schedule of Performance data collections, analysis, and subsequent tuning (as needed) is strongly encouraged. Using the PMAT tool at some regular interval (perhaps monthly) can uncover trends in application throughput on the TBSM Data Server, as new Business Services are added to the in-memory model. Also, as additional Web Consoles are added to the Dashboard Server, looking at such metrics as overall garbage collection pause times might be helpful in uncovering tuning areas to reduce application response times while serving a higher number of end-users.

In summary, making performance analysis a proactive subject in your own unique TBSM environments can go a long way to minimizing or preventing future performance concerns.

Hardware for Production Environments

Java Native Memory considerations

Overview

Java process memory is made up of Heap and Native memory areas. Heap memory is used for instances of class objects, while native memory holds non-instance information such as stack and thread data.

As well as Java heap memory exhaustion, it is possible for Java applications to run out of native memory (even if the heap is not exhausted) - due to exhausting all the memory on the physical server, or hitting a 32-bit process memory limit.

The following is a common error, seen in the Java core file or trace logs, when native memory is exhausted:

Dump Event "systhrow" (00040000) Detail "java/lang/OutOfMemoryError"

"Failed to create a thread: retVal -1073741830, errno 11" received

Recommended tools to check Java memory usage include reviewing /proc/{pid}/VmData and /proc/{pid}/VmStack in Linux, and leveraging the Performance Monitor tool in Windows.

Settings Recommended for TBSM to Reduce OOM Native Memory

-Xmns384m

-Xmnx384m

-Xmos1152m

-Xmox1152m

-Xgcpolicy:gencon

Notes:

Operating Systems Specifics

For General Operating System specific tuning, see this link:

http://publib.boulder.ibm.com/infocenter/wasinfo/fep/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tprf_tuneopsys.html

Linux on System z Performance: Requesting Larger Heaps for 31-Bit Java Kits

If you are running a 31-bit Java kit, you may be able to allocate heap sizes up to 1.6GB using a modified memory layout, depending on the amount of available memory. This is true for 31-bit Java Kits in a 64-Bit Linux environment also. Depending on which version of Linux is running in your environment, you should be able to use one of the two sets of instructions listed below to increase your heap size up to 1.6GB.

Attempting to secure a larger heap size on SLES 9 or SLES 10 – running a Linux kernel version prior to 2.4.19 – involves modifying the setting in the /proc/<PID>/mapped_base file. This file allows one to reorder the mapped base for the shared libraries. However, manipulating the mapped_base setting on a SLES 10 system with a Linux kernal version of 2.4.19 or higher is not necessary or even possible. Architectural changes in later versions of the Linux OS kernel render the mapped_base option obsolete, which will eventually result in a deprecation of the mapped_base function altogether. Therefore, you may request a heap size up to 1.6GB on the later versions of the kernel now.

Users running 31-Bit emulation mode on a Novell SLES9 or SLES10 setup, with a kernel version prior to 2.4.19, may attempt to increase the size of the heap by following these steps.

1. First, determine the process id (PID). PID is the process ID of the process you want to change the layout (usually the bash shell). $$ will give the current shell PID. You can issue the command cat /proc/self/maps to obtain the PID also.

2. Display the current memory layout of the process by issuing the following command: cat /proc/<PID>/maps

3. Check the mapped_base value of the process by issuing the following command: cat /proc/<PID>/mapped_base

4. Lower the mapped_base value by entering the following command: echo 268435456 > /proc/<PID> /mapped_base

5. You may now attempt to increase the size of your heap.

Users running 31-Bit emulation mode on RHEL4 or RHEL 5 distributions may be able to obtain a larger heap size by means of the flex-mmap patch and turning off Linux pre-linking.

1. Determine the current state of the flex-mmap patch on your system by executing the following command: cat /proc/sys/vm/legacy_va_layout

2. Review the response. A response of 0 means flex-mmap is enabled. A response of 1 indicates the old memory layout is active.

3. If flex-mmap is disabled, enable it by executing the following command: echo 0 > /proc/sys /vm/legacy_va_layout

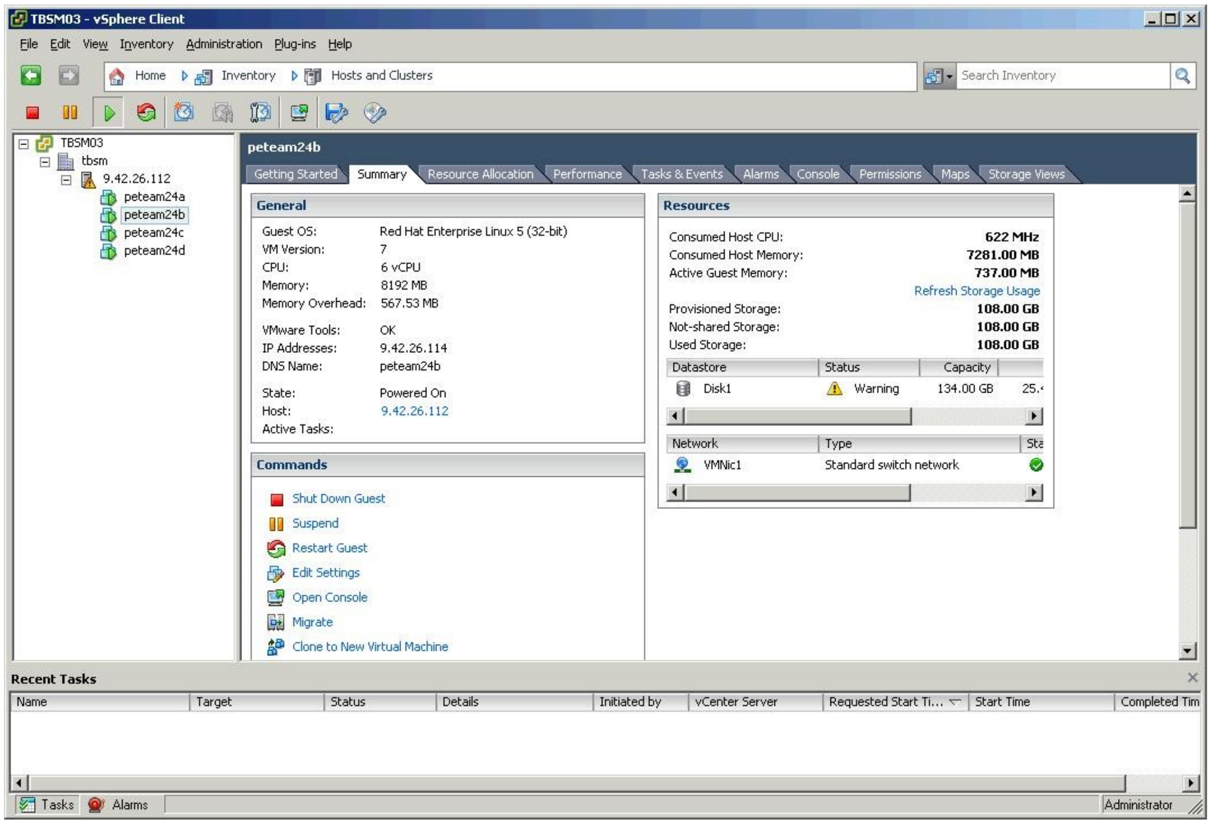
4. Disable Linux pre-linking. In the file /etc/sysconfig/prelink set PRELINKING=no and apply the new setting by running the daily cron prelink job immediately: /etc/cron.daily/prelink and hit <ENTER>

5. You may now attempt to increase the size of your heap.

Virtual Machines

Below are VMWare screencap and settings recommended for use when running TBSM on Virtual Machines.

1. Go to/launch the vSphere Client. Expand the hierarchy on the left to view all the VM's in the environment:



General

Guest OS:	Red Hat Enterprise Linux 5 (32-bit)
VM Version:	7
CPU:	6 vCPU
Memory:	8192 MB
Memory Overhead:	567.53 MB
VMware Tools:	OK
IP Addresses:	9.42.26.114
DNS Name:	peteam24b
State:	Powered On
Host:	9.42.26.112
Active Tasks:	

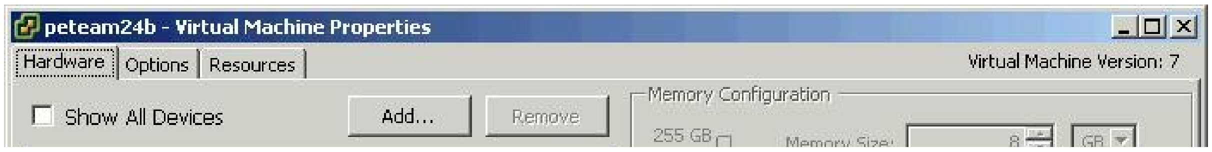
Resources

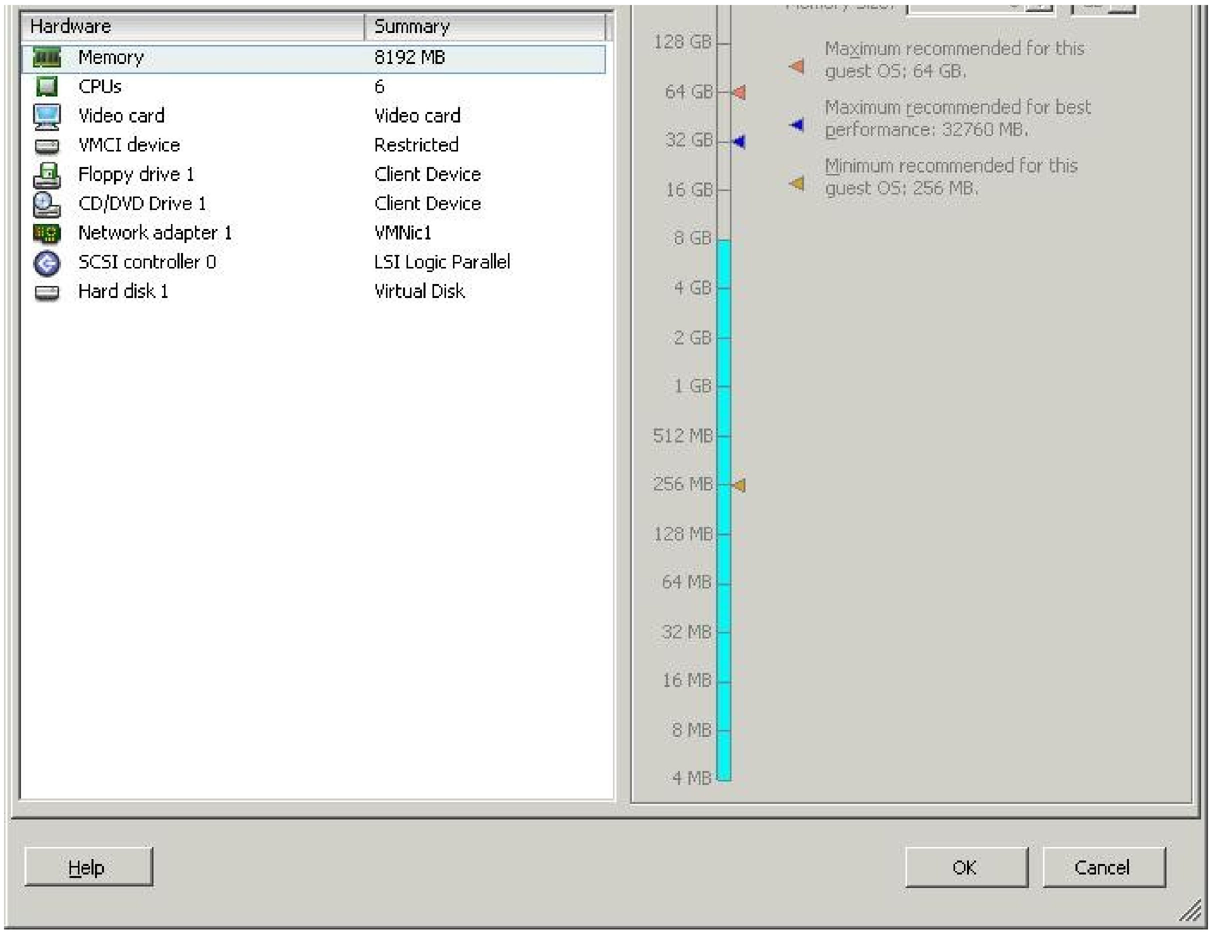
Consumed Host CPU:	622 MHz
Consumed Host Memory:	7281.00 MB
Active Guest Memory:	737.00 MB
	Refresh Storage Usage
Provisioned Storage:	108.00 GB
Not-shared Storage:	108.00 GB
Used Storage:	108.00 GB

Datastore	Status	Capacity	
Disk1	Warning	134.00 GB	25.4

Network	Type	Sta
VMNic1	Standard switch network	✓

2. Right-click on the Virtual Machine > Left-click "Edit Settings" from the pop-up to get here:

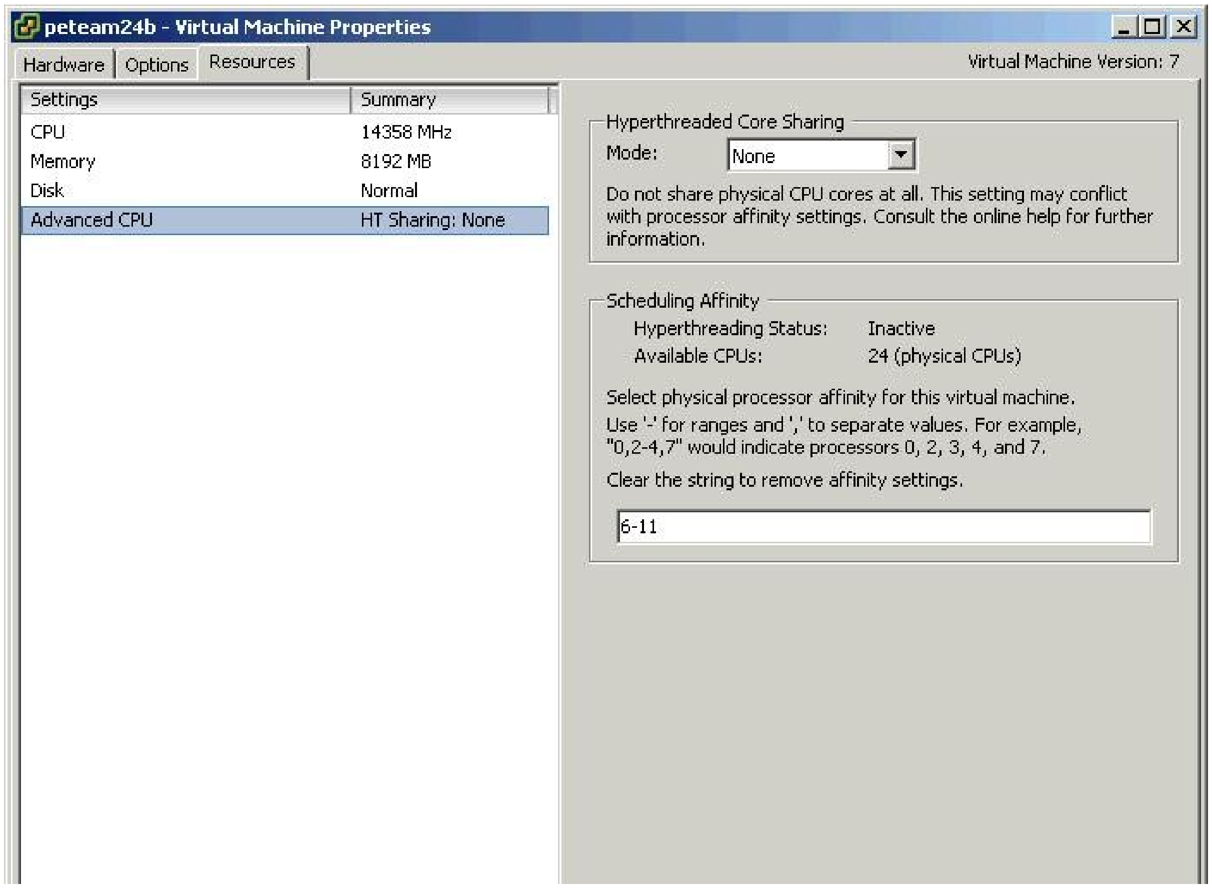




Hardware	Summary
Memory	8192 MB
CPUs	6
Video card	Video card
VMCI device	Restricted
Floppy drive 1	Client Device
CD/DVD Drive 1	Client Device
Network adapter 1	VMNic1
SCSI controller 0	LSI Logic Parallel
Hard disk 1	Virtual Disk

Maximum recommended for this guest OS: 64 GB.
Maximum recommended for best performance: 32760 MB.
Minimum recommended for this guest OS: 256 MB.

3. Click on the Resources tab (from the above panel) and select "Memory"... to RESERVE the 8 GB of memory, move the slider for "Reservation" to the right... it should look something like this when you are done:



Virtual Machine Version: 7

Settings	Summary
CPU	14358 MHz
Memory	8192 MB
Disk	Normal
Advanced CPU	HT Sharing: None

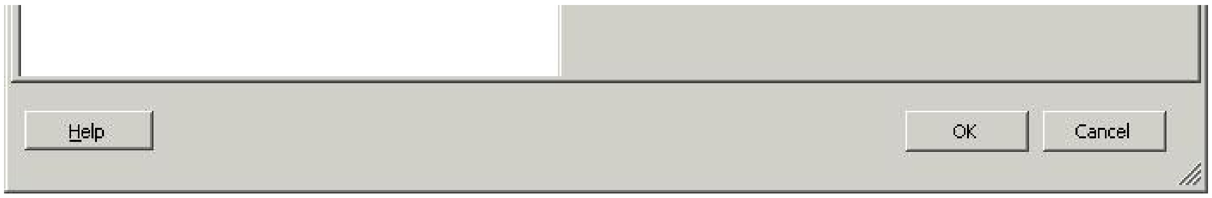
Hyperthreaded Core Sharing
Mode: None

Do not share physical CPU cores at all. This setting may conflict with processor affinity settings. Consult the online help for further information.

Scheduling Affinity
Hyperthreading Status: Inactive
Available CPUs: 24 (physical CPUs)

Select physical processor affinity for this virtual machine.
Use '-' for ranges and ',' to separate values. For example, "0,2-4,7" would indicate processors 0, 2, 3, 4, and 7.
Clear the string to remove affinity settings.

6-11



4. And for "hard-coding" the CPU to a specific JVM (this is helpful when comparing vmstat/sar/top data from the VM to esxtop from the ESX Server), click on the "Advanced CPU" selection:

Turn hyperthreading off and for "Scheduling Affinity", select the cores you want to assign to that specific Virtual Machine. After making these changes, reboot the VM.

References,Trademarks, and Notices

References

1. TBSM 4.2 Beta Web Conference Series: Performance Tuning: Internal IBM presentation delivered in September 2008 to customers participating in the TBSM 4.2 beta program
2. PostgreSQL Online Documentation: http://www.postgresql.org/docs/8.0/static/index.html
3. Tivoli Business Service Manager 4.2 Installation and Administrator's Guides: http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/
4. A reference book for everything related to IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 5.0. (In PDF format.):http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/diagnosis/diag50.pdf
5. Tuning Garbage Collection with the 5.0 Java Virtual Machine: http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html
6. Linux on System z Performance Update: http://linuxvm.org/present/SHARE113/S2190mh.pdf
7. IBM Pattern Matching and Analysis (PMAT) tool from IBM Alphaworks: http://www.alphaworks.ibm.com/tech/pmat
8. How to tell if an OOM is from native or heap memory exhaustion: http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.java/java/6.0/MonitorDebug/1-IntroAndDetermining/player.html
9. How to check what memory is used on different operating systems: http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.java/java/6.0/MonitorDebug/2-MonitoringNativeMemoryUsage/player.html
10. How to trace allocations or native memory: http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.java/java/6.0/MonitorDebug/3-ProfilingNativeMemoryUsage/player.html
11. Troubleshooting native memory issues: http://www.ibm.com/support/docview.wss?rs=180&uid=swg21373312
12. Native Memory information: http://www.ibm.com/developerworks/java/library/j-nativememory-linux/
13. The following also gives a useful explanation of Native Memory usage and Process Address Space for different Operating Systems: http://publib.boulder.ibm.com/infocenter/javasdk/tools/index.jsp?topic=%2Fcom.ibm.java.doc.igaa%2F_1vg000121410cbe-1195c23a635-7ffa_1001.html

Trademarks

Copyright and Notices

TBSM Databases for Service Model Storage

A PostgreSQL database can be a very fast database, but the as-is configuration tends to be rather conservative. A few configuration changes to the postgresql.conf file can improve
PostgreSQL performance dramatically. Note that these settings worked well in the performance test environment, and are provided as a starting point for your own unique environments.

Important: Back up your original postgresql.conf file before making any changes.

Shared_buffers: Sets the number of shared memory buffers that are used by the database server. The default is typically 1000 X 8K pages. Settings significantly higher than the minimum are usually needed for good performance; values of a few thousand are recommended for production installations. This option can only be set at server start-up.

Suggestion: shared_buffers = 16384

If editing this setting, also change the rad_dbconf file pg_buffer parameter in UNIX or Linux systems to the same value.

Work_mem: Non-shared memory that is used for internal sort operations and hash tables. This setting is used to put a limit on any single operation memory-utilization before being forced to use disk.

Suggestion: work_mem = 32000

Effective_cache_size: Sets the planner's assumption about the effective size of the disk cache that is available to a single index scan. This is factored into estimates of the cost of
using an index; a higher value makes it more likely that index scans are used, a lower value makes it more likely sequential scans are used.

Suggestion: effective_cache_size = 30000

Random_page_cost: Sets the planner's estimate of the cost of a nonsequentially fetched disk page. This is measured as a multiple of the cost of a sequential page fetch. A higher value makes it more likely that a sequential scan is used, a lower value makes it more likely an index scan is used.

Suggestion: random_page_cost = 2

Fsync: To speed up bulk loads by way of the XML Toolkit, disable the fsync parameter in the postgresql.conf file as follows:

fsync = false # turns forced synchronization on or off

The fsync parameter sets whether you write data to disk as soon as it is committed, which is done through the Write Ahead Logging (WAL) facility. Do this only if you want faster load times; the caveat is that the load scenario mighty need to run again if the server shuts down prior to the completion of processing due to a power failure, disk crash, and so on.

After changing a large amount of data in Postgres, you should vacuum the TBSM Data Server database to improve performance. A vacuumdb utility is provided with the PostgreSQL database that can be used to clean up database storage. Running this utility periodically or after a significant number of database rows change helps subsequent queries process more efficiently. The utility resides in the $TBSM_HOME/platform/<arch>/pgsql8/bin/vacuumdb directory and can be run as follows:

Unix: $TBSM_HOME/platform/arch/pgsql8/bin/vacuumdb -f -z -p 5435 -U postgres rad

Windows: %TBSM_HOME%\pgsql8\bin\vacuumdb -f -z -p 5435 -U postgres rad

The parameters for the vacuumdb command:

-f: The utility does a full vacuum
-z: The utility analyzes and updates statistics that are used by the query planner
5435: The port that the database process is listening on
postgres: The user ID used to connect to the rad database
rad: The database name

The only output expected from the command is the word "VACUUM".

Important: The TBSM Discovery Library toolkit periodically vacuums the tables that are used by the toolkit. Control of this is handled with the DL_DBVacuum properties in the xmltoolkitsvc.properties file. For more information on these properties, see the Discovery Library toolkit properties. Depending on how often the toolkit imports data, the automatic vacuums might be sufficient.

As some SLA tracking data accumulates over time, the tables in IBM Tivoli Business Service Manager's database that store this tracking data grow. This can lead to dramatically reduced performance in event processing, model configuration changes, and SLA tracking. Below are some guidelines for maintaining the SLA tracking data at a reasonable size:

There are essentially 7 tables used to store SLA tracking data as follows:

Duration tracking

durationcountwatchers

tasksofdurationcountwatchers

Incident Tracking

incidentcountwatchers

tasksofincidentcountwatchers

Cumulative (overtime) duration tracking

cumuldurationrulestate

cumulrulearchive

plus radeventstore, which is an additional table which stores event references for SLA trackers.

These tables are used for resyncing SLA status when the TBSM server is restarted. In addition, the tables for the cumulative trackers are used to provide the database for the historical SLA graphs.

The best strategy for preventing these tables from getting too big is to periodically delete from them all entries older than a specified amount. The smaller the time-window of entries left in the database, the less likely it will be to run into associated performance issues. However it is means that SLA charts will only be able to go back that far in history.

The Postgres SQL command to do this is as follows:

delete from <tablename> where timestamp < (select date_part('epoch',now()) - 3600 * 24 * <numdays>) * 1000;

So to delete all entries from the durationcountwatchers table that are more than 30 days old the command would be:

delete from durationcountwatchers where timestamp < (select date_part('epoch', now()) - 3600 * 24 * 30) * 1000;

The one exception to this is for the radeventstore table, where the field name is eventtime instead of timestamp. So to delete all entries older than 30 days in the radeventstore table, the command is:

delete from radeventstore where eventtime < (select date_part('epoch',now()) - 3600 * 24 * 30) * 1000;

Note: The following APARs, shipped with 4.2.1-TIV-BSM-FP0003 prevent the following SLA tables from growing too large:

IZ95629 tasksofincidentcountwatchers

tasksofdurationcountwatchers

IV00640 incidentcountwatchers

The TBSM 6.1 DB2 database stores all the information for the TBSM service model. This data also includes templates, policies and rules that determine how the service model changes in relation to data gathered from OMNIbus and other external data sources. Related databases in this release also include tables for the metric and marker data used in the Time Window Analyzer (TWA) portlet.

For 32-bit systems, a minimum of 512 MB of RAM per CPU is recommended, up to 4 GB per machine, to support the buffer pools, DB2 agents, and other shared memory objects required. Typically, a 2-core processor @ 3.0 GHz or greater should be utilized for optimal performance.

The DB2 "self-tuning" memory feature simplifies the task of memory configuration by automatically setting values for memory configuration parameters and sizing buffer pools. When enabled (the typical setting), the memory tuner dynamically distributes available memory resources among the following memory consumers: buffer pools, locking memory, package cache, and sort memory.

For TBSM 6.1 Performance testing, the results using this feature provided extremely reliable system performance and stability, and use of this feature is strongly recommended. For in-depth review, please refer to:

http://publib.boulder.ibm.com/infocenter/db2luw
/v9r7/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.perf.doc%2Fdoc%2Fc0021627.html

Distribute TBSM databases to separate logical disk volumes to distribute I/O overhead and remove disk contention (this is especially useful if the Time Window Analyzer feature of TBSM will be leveraged).

In addition, leverage an additional (unshared) logical disk volume for DB2 transaction logs to minimize any transaction logging overhead.

For example, on a Windows-based DB2 system comprised of multiple logical disk volumes, the TBSM databases might be distributed as such:

D:\DB2\TBSM \\ TBSM Database

E:\DB2\TBSMMM \\ TWA Metric & Marker Database

F:\DB2\TBSMHIST \\ TWA Metric History Database

G:\DB2\db2_logs \\ DB2 transaction logs

The Discovery Library toolkit is a component withi**n TBSM. It typically resides on the TBSM D**ata server and accesses SCR tables in the TBSM database. This set of SCR tables is added to the TBSM database as part of the Discovery Library toolkit installation. TBSM uses an ESDA rule to query these tables and creates new services based on the SCR data. The toolkit is composed of a service/daemon that monitors the data source and adds the data to the SCR tables.

For reference, the Service Component Repository (SCR): TBSM stores a subset of service data from the Common Data Model in a set of Service Component Repository (SCR) tables. These tables are filtered by a set of SCR service templates that define the common behavior for a given service type.These SCR service templates automatically can create service models based on the data in the SCR tables.

Of performance interest to Customers: leave the Discovery Library Adapter (XML Toolkit) running, as there is a thread that runs nightly to clean up the SCR tables in DB2.

In relation to the SCR maintenance thread, Customers may wish to consider running the following DB2 command on their TBSM databases during off-hours (as a scheduled job perhaps) to update the update the statistics used by the DB2 optimizer:

“reorgchck update statistics on table all”

As noted in the DB2 InfoCenter, "Do not tune just for the sake of tuning: Tune to relieve identified constraints". Tuning resources that are not the primary cause of performance problems can actually make subsequent tuning work more difficult.

As earlier mentioned, leveraging the Self-tuning memory function of DB2 9.7, in combination with the distribution of TBSM databases and transaction logging, removed any disk or I/O concerns.

For Customers who require additional insights into DB2 tuning, the following website is recommended:

http://pic.dhe.ibm.com/infocenter/db2luw
/v9r7/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.perf.doc%2Fdoc%2Fc0005414.html

WebSphere Application Server Heap Memory Tuning

Overview

The TBSM 6.1 release leverages an embedded version of the WebSphere Application Server 7.0 for the Data Server and Dashboard Server components. Prior TBSM 4.2.x releases use older, embedded versions of WebSphere 6.1. However, common across ***all*** release levels is the core concept pertaining to efficiently tuning the WebSphere server's JVM (Java Virtual Machine) for the TBSM components.

Note that there are several Operating System tuning recommendations provided by the IBM WebSphere team. Prior to any tuning, please review the following content for your specific Operating System(s) and configure as per the recommendations:

IBM WebSphere Application Server, Operating System Tuning Steps

Following the Operating System tuning steps from WebSphere, the recommended tuning steps for TBSM include:

1. Identify the current TBSM JVM Memory settings

2. Enable JVM Garbage Collection (GC) logging

3. Run a representative workload

4. Analyze the GC log results

5. Tune the JVM Memory appropriately

6. Review the new results

The following statements are taken from the WebSphere 6.1 documentation in reference to Java memory and heap tuning:

"The JVM memory management and garbage collection functions provide the biggest opportunities for improving JVM performance."

and:

"Garbage collection normally consumes from 5% to 20% of total execution time of a properly functioning application. If not managed, garbage collection is one of the biggest bottlenecks for an application."

The TBSM Data Server and Dashboard Server(s) each run in a separate JVM; subsequently, each has the capability to be independently tuned. Of primary consideration is the memory allocation to each of the JVMs, bounded by two key values:

• Initial memory (initialHeapSize)

• Maximum memory (maximumHeapSize)

For reference, the Data Server and Dashboard Server each use the default garbage collection algorithm (optthruput) which can be used without modification (with the exception of the Solaris Operating Environment, which uses a generational garbage collector instead). The following statement is from the WebSphere 6.1 documentation:

"optthruput, which is the default, provides high throughput but with longer garbage collection pause times. During a garbage collection, all application threads are stopped for mark, sweep and compaction, when compaction is needed. optthruput is sufficient for most applications."

Based on TBSM performance analysis, this default algorithm has proven quite capable, and is recommended in for Customer use cases including:

- · Less than 25,000 Service Instances
- · Less than 25 concurrent Web Consoles

For Customers with larger deployments, the generational garbage collection algorithm (gencon) would be recommended. For an example of the gencon settings, please see the **" Settings recommended for TBSM to reduce OOM Native Memory**" section later in this document.

(For reference on the default Oracle Garbage collection algorithm, review the Oracle JVM link provided in the **Reference** section of this document.)

The sections which follow will explain how to efficiently size the TBSM JVMs to allow the default garbage collection algorithms to most operate most efficiently. Note: The TBSM references which follow are valid for both the TBSM 4.2.x and 6.1 levels of code.

To determine the Java version and level that is in use, run the following command:

$TIP_HOME/java/bin/java -version

In response to this command, the TBSM server writes information to the command line, including the JVM provider information and level of release. Knowing this up-front directs you to the correct parameters that follow in this document for Java™ Virtual Machine configuration.

A few considerations about JVM sizing and GC activity:

Proper JVM heap memory sizing is critical to TBSM.

Memory is allocated to object storage within the JVM heap; as the number of objects grows, the amount of free space within the heap decreases. When the JVM cannot allocate additional memory for new objects as it nears it's upper memory threshold (Xmx value), a Garbage Collection (GC) is called by the JVM to reclaim memory from objects no longer referenced to satisfy this request.

Depending on the JVM and type of GC activity, this garbage collection processing can temporarily suspend other threads in the JVM, granting the garbage collection thread(s) priority to complete the GC work as quickly and efficiently as possible. This prioritization of GC threads and pausing of the JVM is commonly referred to as a "Stop the World" pause for the optthruput algorithm. With proper heap analysis and subsequent JVM tuning, this overhead can be minimized, thereby increasing TBSM application throughput. Essentially, the JVM spends less time paused for GC activities, and more time processing core TBSM activities.

1. Identify the Current JVM Memory Settings for TBSM

There are several ways to gather (and tune) the WebSphere JVM settings in a TBSM environment. One of the easiest (and safest) ways to do this is by using setJVMSettings script in the tbsm/bin directory, as follows:

Data Server

Unix: $TBSM_HOME/bin/setJVMSettings.sh tipadmin <password> data -showJVMSettings

Windows: %TBSM_HOME%\bin\setJVMSettings.bat tipadmin <password> data -showJVMSettings

Dashboard Server

Unix: $TBSM_HOME/bin/setJVMSettings.sh tipadmin <password> dashboard -showJVMSettings

Windows: %TBSM_HOME%\bin\setJVMSettings.bat tipadmin <password> dashboard -showJVMSettings

Note: setJVMSettings is available in service release 4.2.1-TIV-BSM-IF0004, or available for download for earlier releases.

Alternatively, look at the server.xml file for each of the servers. This file is located as follows:

Data Server

Unix: $TIP_HOME/profiles/TBSMProfile/config/cells/TBSMCell/nodes/TBSMNode/servers/server1/server.xml

Windows: %TIP_HOME%\profiles\TBSMProfile\config\cells\TBSMCell\nodes\TBSMNode\servers\server1
\server.xml

Dashboard Server

Unix: $TIP_HOME/profiles/TIPProfile/config/cells/TIPCell/nodes/TIPNode/servers/server1/server.xml

Windows: %TIP_HOME%\profiles\TIPProfile\config\cells\TIPCell\nodes\TIPNode\servers\server1\server.xml

Important Notes:

· Before making any changes to the server.xml file, ensure a backup copy of the file is preserved, and all changes have been tested on a non-production server.

· The server.xml file should only be modified by the user who installed TBSM.

The following section is part of the default server.xml file from the TBSM Data Server:

<jvmEntries xmi:id="JavaVirtualMachine_1273090887841" verboseModeClass="false"

*verboseModeGarbageCollection="false" verboseModeJNI="false" initialHeapSize="**256**" maximumHeapSize="**512**" runHProf="false" hprofArguments="" debugMode="false" debugArgs="-Djava.compiler=NONE -Xdebug -Xnoagent -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=16050" genericJvmArguments="-Xdisableexplicitgc">*

Note for TBSM 4.2.x, the values set for initialHeapSize and maximumHeapSize are "256" and "512". These 2 settings are responsible for setting the initial JVM size (initialHeapSize) to 256 MB of memory, and the maximum JVM size (maximumHeapSize) to 512 MB of memory. For TBSM 6.1, the default maximumHeapSize is set to 1200 MB of memory.

2. Enable Java Virtual Machine (JVM) Garbage Collection (GC) Logging

There are several ways to log garbage collection (GC) data for later analysis. One of the easiest (and safest) ways to do this is by using setJVMSettings script in the tbsm/bin directory, as follows:

Data Server

Unix: $TBSM_HOME/bin/setJVMSettings.sh tipadmin <password> data -addDefaultGCLogging

Windows: %TBSM_HOME%\bin\setJVMSettings.bat tipadmin <password> data -addDefaultGCLogging

Dashboard Server

Unix: $TBSM_HOME/bin/setJVMSettings.sh tipadmin <password> dashboard -addDefaultGCLogging

Windows: %TBSM_HOME%\bin\setJVMSettings.bat tipadmin <password> dashboard -addDefaultGCLogging

Important Notes:

· A server re-start is required for these changes to take effect.

· Non-default GC collection settings can be enables using the –setJVMArguments switch.

· setJVMSettings is available in service release 4.2.1-TIV-BSM-IF0004, or available for download for earlier releases.

Alternatively, garbage collection (GC) can be enabled by editing the server.xml file, as follows:

1. Update the value for verboseModeGarbageCollection from false to true

2. Add the following arguments to the genericJvmArguments setting in server.xml:

Data Server

Unix: -Xverbosegclog:${SERVER_LOG_ROOT}/verboseDataGC.%Y%m%d.%H%M%S.%pid.txt

Windows: -Xverbosegclog:${SERVER_LOG_ROOT}\verboseDataGC.%Y%m%d.%H%M%S.%pid.txt

Dashboard Server

Unix: -Xverbosegclog:${SERVER_LOG_ROOT}/verboseDashGC.%Y%m%d.%H%M%S.%pid.txt

Windows: -Xverbosegclog:${SERVER_LOG_ROOT}\verboseDashGC.%Y%m%d.%H%M%S.%pid.txt

For example:

Unix:

<jvmEntries xmi:id="JavaVirtualMachine_1273090887841" verboseModeClass="false" verboseModeGarbageCollection="true" ... genericJvmArguments="-Xdisableexplicitgc -Xverbosegclog:${SERVER_LOG_ROOT}/verboseDataGC.%Y%m%d.%H%M%S.%pid.txt">

Windows:

<jvmEntries xmi:id="JavaVirtualMachine_1273090887841" verboseModeClass="false" verboseModeGarbageCollection="true" ... genericJvmArguments="-Xdisableexplicitgc -Xverbosegclog:${SERVER_LOG_ROOT}\verboseDataGC.%Y%m%d.%H%M%S.%pid.txt">

Note that the directory for the GC logfile must exist prior to launching TBSM with the updated server.xml file. For this scenario, the server variable ${SERVER_LOG_ROOT} is used, which evaluates to <TIP_HOME>/profiles /TIPProfile/logs/server or TBSMProfile/logs/server1 directory, depending on which server is updated. Whatever directory is used, the TBSM install user must have read and write permissions for the location. The log file names should be different both to distinguish between the two, especially when both TBSM Data and Dashboard Servers reside on the same system.

Operating System specifics

On Solaris or HP-UX Operating Systems, the following options are also required:

-XX:+PrintGCTimeStamps

-XX:+PrintGCDetails

-XX:+PrintHeapAtGC

For example:

<jvmEntries xmi:id="JavaVirtualMachine_1273090887841" verboseModeClass="false" verboseModeGarbageCollection="true" ... genericJvmArguments="-Xdisableexplicitgc -XX:+PrintGCTimeStamps -XX:+PrintGCDetails -XX:+PrintHeapAtGC -Xverbosegclog:${SERVER_LOG_ROOT}/verboseDataGC.%Y%m %d.%H%M%S.%pid.txt">

3. Run a Representative Workload

At this point, start the TBSM servers (Data Server first, then Dashboard Server as soon as the processor quiesces on the Data Server). Next, proceed with a common scenario or representative workload in your environment to populate the GC logs for subsequent performance analysis. This can be a simple scenario that you would like to optimize, such as TBSM Data Server start-up, or perhaps a steady-state workload for 30 minutes, as in the following example.

First, record some notes on the TBSM environment configuration. The following scenario was measured for initial performance and subsequent tuning:

Service Model: 5000 Service Instances, 4 level hierarchy, no leaf node with more than 50 children.

· Initial heap size: 256 MB

· Maximum heap size: 512 MB

· Data Server Started: 9:57:00

· Dashboard Server Started: 9:59:00

· Workload Start Time: 10:09:00

· Workload End Time: 10:39:00

For this reference scenario, the TBSM Data Server was started with GC logging at 9:57:00. After the processor quiesces on the server (indicating that the Data Server start-up and initial Service Model processing had completed), the Dashboard Server was started at 9:59:00 and 50 unique TBSM Web Consoles were logged in; each Web Console was configured to use a unique Service Tree and Service Viewer perspective.

Finally, a steady-state event workload using thousands of unique events (sent via remote Tivoli EIF probes) was introduced at 10:09:00, and continued until 10:39:00 when the event flow was stopped and GC log files were immediately collected.

Also, while this workload was being processed, a "vmstat -n 15 120 >> vmstat_out.txt" command was run (on each TBSM server) to collect CPU statistics every 15 seconds for 30 minutes to a local file for later analysis. After the workload was complete, these vmstat_out.txt files were also collected for review.

4. Analyze the GC Log Results

To analyze the resultant GC log files, download the IBM Pattern Matching and Analysis (PMAT) tool from the IBM Alphaworks Web site:

http://www.alphaworks.ibm.com/tech/pmat

Note that this tool is typically updated over time, and a good practice is to check for updates prior to beginning a new series of performance studies.

Taken from the PMAT Web site:

"The Pattern Modeling and Analysis Tool for IBM® Java Garbage Collector (PMAT) parses verbose GC trace, analyzes Java heap usage, and recommends key configurations based on pattern modeling of Java heap usage... This information can be used to determine whether garbage collections are taking too long to run; whether too many garbage collections are occurring; and whether the JVM crashed during garbage collection."

Although there is an in-depth tutorial on the same Web site (See: Webcast replay - ["How to analyze verbosegc trace with IBM Pattern Modelling and Analysis Tool for IBM Java Garbage Collector" (http://www-01.ibm.com /support/docview.wss?uid=swg27007240&aid=1), the following information is provided to expedite utilization of the PMAT tool within a Windows environment.

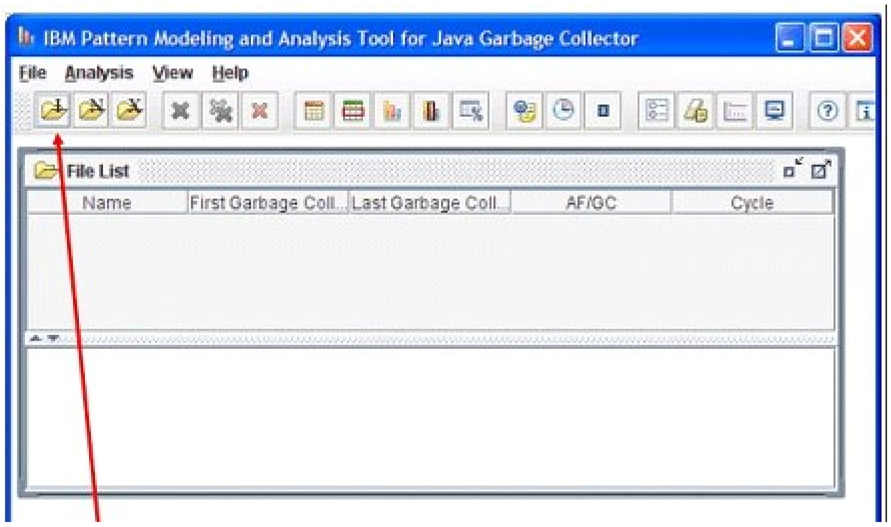
To analyze the GC log file that you collected, start the IBM PMAT Tool:

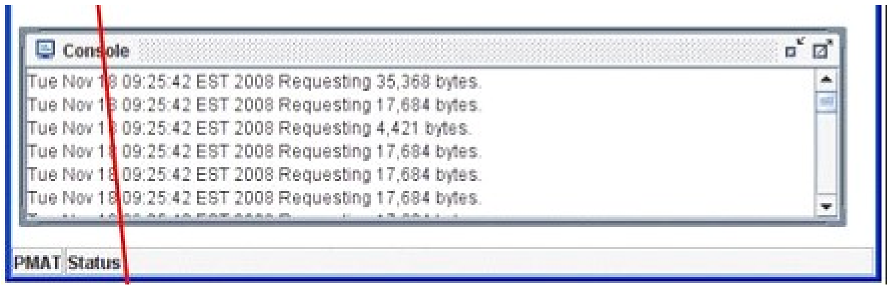
"C:\Program Files\Java\jdk1.6.0\bin\java" -Xmx128m -jar ga433.jar"

Use this example and edit it as needed (substitute the location of your Java executable file and location of PMAT files). Note that the Xmx value of 128m limits the PMAT tool to use no more than 128 MB RAM on the system. If you have a number of very large GC log files, you might want to increase the Xmx value.

Review the PMAT Web site for other configuration details or a more in-depth walk-thru as needed. The following examples assume that the tool is correctly installed and ready for you to use.

The following screen capture shows the initial screen of the PMAT tool.





```
Tue Nov 18 09:25:42 EST 2008 Requesting 35,368 bytes.
Tue Nov 18 09:25:42 EST 2008 Requesting 17,684 bytes.
Tue Nov 18 09:25:42 EST 2008 Requesting 4,421 bytes.
Tue Nov 18 09:25:42 EST 2008 Requesting 17,684 bytes.
Tue Nov 18 09:25:42 EST 2008 Requesting 17,684 bytes.
Tue Nov 18 09:25:42 EST 2008 Requesting 17,684 bytes.
```

Click the I folder to load an IBM generated GC log; the IBM version is used across all TBSM-supported Operating Systems with the exception of the Solaris Operating environment which uses the Oracle JVM.

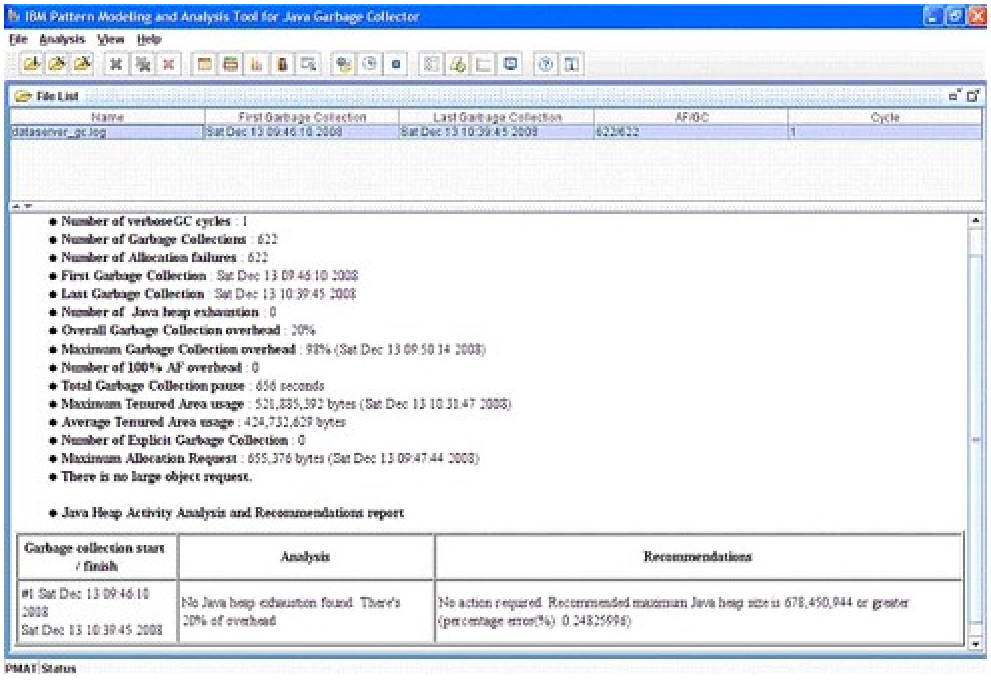
To open a Oracle-generated log, click the N folder instead. The remainder of this document assumes an IBM-generated log is used for the 30 minute steady-state scenario.

Navigate to the GC log that you want to analyze and select it. The PMAT tool processes the log, and displays the analysis and recommendations you can review.

5. Tune the JVM Memory

Data Server

The following screen capture shows the result after a garbage collection log has been opened within the PMAT tool for the TBSM Data Server.


IBM Pattern Modeling and Analysis Tool for Java Garbage Collector
File
Analysis
View
Help
File List
PMAT Status

Name	First Garbage Collection	Last Garbage Collection	AF/GC	Cycle
dataserver_gc.log	Sat Dec 13 09:46:10 2008	Sat Dec 13 10:39:45 2008	622/622	1

- **Number of verboseGC cycles** : 1
- **Number of Garbage Collections** : 622
- **Number of Allocation failures** : 622
- **First Garbage Collection** : Sat Dec 13 09:46:10 2008
- **Last Garbage Collection** : Sat Dec 13 10:39:45 2008
- **Number of Java heap exhaustion** : 0
- **Overall Garbage Collection overhead** : 20%
- **Maximum Garbage Collection overhead** : 98% (Sat Dec 13 09:50:14 2008)
- **Number of 100% AF overhead** : 0
- **Total Garbage Collection pause** : 656 seconds
- **Maximum Tenured Area usage** : 521,885,392 bytes (Sat Dec 13 10:31:47 2008)
- **Average Tenured Area usage** : 424,732,629 bytes
- **Number of Explicit Garbage Collection** : 0
- **Maximum Allocation Request** : 655,376 bytes (Sat Dec 13 09:47:44 2008)
- **There is no large object request.**

- **Java Heap Activity Analysis and Recommendations report**

Garbage collection start / finish	Analysis	Recommendations
#1 Sat Dec 13 09:46:10 2008 Sat Dec 13 10:39:45 2008	No Java heap exhaustion found. There's 20% of overhead	No action required. Recommended maximum Java heap size is 678,450,944 or greater (percentage error(%): 0.24825998)

Review the Analysis and Recommendations sections. For this scenario, the Analysis section indicates that no Java heap exhaustion was found, typically indicating that there is sufficient space within the JVM to satisfy required memory allocations. However, the Overall Garbage Collection Overhead metric notes that 20% of the application time was spent performing garbage collection activities, most likely indicating a need for tuning the JVM memory parameters.

To minimize the GC overhead, review the Recommendations section and assign additional memory to the JVM for more efficient processing of the workload. As the PMAT tool recommendation is to set the JVM Xmx value to approximately 678 MB or greater (and because the system has plenty of memory), a new value of 1024 MB was chosen as the new maximumHeapSize value (recall that the as-provided maximumHeapSize is set to 512 MB in TBSM 4.2.x).

Notes:

· In 32-bit environments, the maximumHeapSize should never exceed 1536 MB

· The minimumHeapSize should be set to the same as the maximumHeapSize

To make this change using setJVMSettings, do the following:

Data Server

Unix: $TBSM_HOME/bin/setJVMSettings.sh tipadmin <password> data -setJVMHeapSize 1024 1024

Windows: %TBSM_HOME%\bin\setJVMSettings.bat tipadmin <password> data -setJVMHeapSize 1024 1024

Notes:

· A server re-start is required for these changes to take effect

· setJVMSettings is available in service release 4.2.1-TIV-BSM-IF0004, or available for download for earlier releases

Alternatively, the server.xml file for the server can be edited:

1. Save a copy, then open the server.xml file for editing

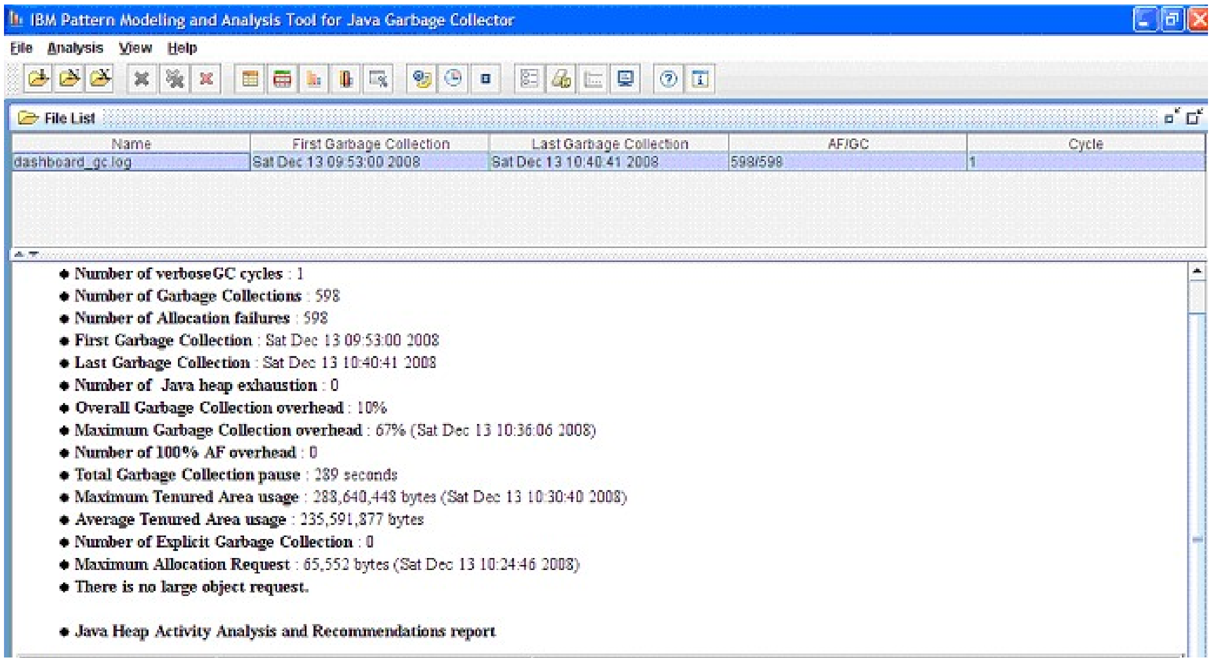
2. Change the maximumHeapSize value from maximumHeapSize="512" to maximumHeapSize="1024"

3. Change the minimumHeapSize value to the same value as the maximumHeapSize value, i.e. minimumHeapSize="1024"

4. Save the changes to the file.

Dashboard Server

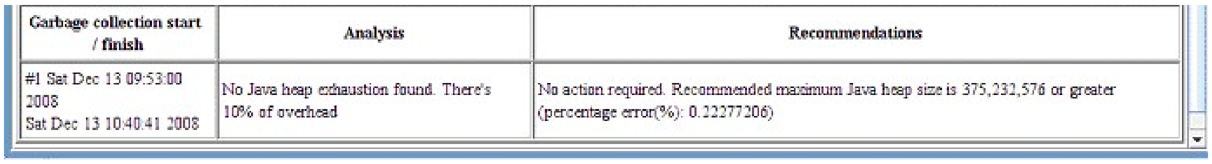
The following screen capture shows the result after a garbage collection log has been opened within the PMAT tool for the TBSM Dashboard Server.


IBM Pattern Modeling and Analysis Tool for Java Garbage Collector
File
Analysis
View
Help
File List

Name	First Garbage Collection	Last Garbage Collection	AF/GC	Cycle
dashboard_gc.log	Sat Dec 13 09:53:00 2008	Sat Dec 13 10:40:41 2008	598/598	1

- **Number of verboseGC cycles** : 1
- **Number of Garbage Collections** : 598
- **Number of Allocation failures** : 598
- **First Garbage Collection** : Sat Dec 13 09:53:00 2008
- **Last Garbage Collection** : Sat Dec 13 10:40:41 2008
- **Number of Java heap exhaustion** : 0
- **Overall Garbage Collection overhead** : 10%
- **Maximum Garbage Collection overhead** : 67% (Sat Dec 13 10:36:06 2008)
- **Number of 100% AF overhead** : 0
- **Total Garbage Collection pause** : 289 seconds
- **Maximum Tenured Area usage** : 288,640,448 bytes (Sat Dec 13 10:30:40 2008)
- **Average Tenured Area usage** : 235,591,877 bytes
- **Number of Explicit Garbage Collection** : 0
- **Maximum Allocation Request** : 65,552 bytes (Sat Dec 13 10:24:46 2008)
- **There is no large object request.**

- **Java Heap Activity Analysis and Recommendations report**



Garbage collection start / finish	Analysis	Recommendations
#1 Sat Dec 13 09:53:00 2008 Sat Dec 13 10:40:41 2008	No Java heap exhaustion found. There's 10% of overhead	No action required. Recommended maximum Java heap size is 375,232,576 or greater (percentage error(%): 0.22277206)

For the Dashboard, the Analysis section indicates that no heap exhaustion was found. It also reveals that 10% of the application time was spent performing garbage collection activities, certainly not excessive, but some slight tuning might be beneficial.

To reduce the GC overhead for the Dashboard Server, again review the Recommendations section. As the PMAT tool advises a maximum JVM size of approximately 375 MB or greater (and the TBSM default is already at 512 MB), a change might not be warranted. However, because the system has plenty of memory, an interesting decision is to choose 768 MB as the new maximumHeapSize value, with a new initial size (minimumHeapSize) of 384 MB.

Notes:

· In 32-bit environments, the maximumHeapSize should never exceed 1536 MB

· The minimumHeapSize should be set to the same as the maximumHeapSize

To make this change using setJVMSettings, do the following:

Dashboard Server

Unix: $TBSM_HOME/bin/setJVMSettings.sh tipadmin <password> dashboard -setJVMHeapSize 768 768

Windows: %TBSM_HOME%\bin\setJVMSettings.bat tipadmin <password> dashboard -setJVMHeapSize 768 768

Notes:

· A server re-start is required for these changes to take effect

· setJVMSettings is available in service release 4.2.1-TIV-BSM-IF0004, or available for download for earlier releases

Alternatively, the server.xml file for the server can be edited:

1. Save a copy, then open the server.xml file for editing

2. Change the maximumHeapSize value from maximumHeapSize="512" to maximumHeapSize="768"

3. Change the minimumHeapSize value to the same value as the maximumHeapSize value, i.e. minimumHeapSize="768"

4. Save the changes to the file.

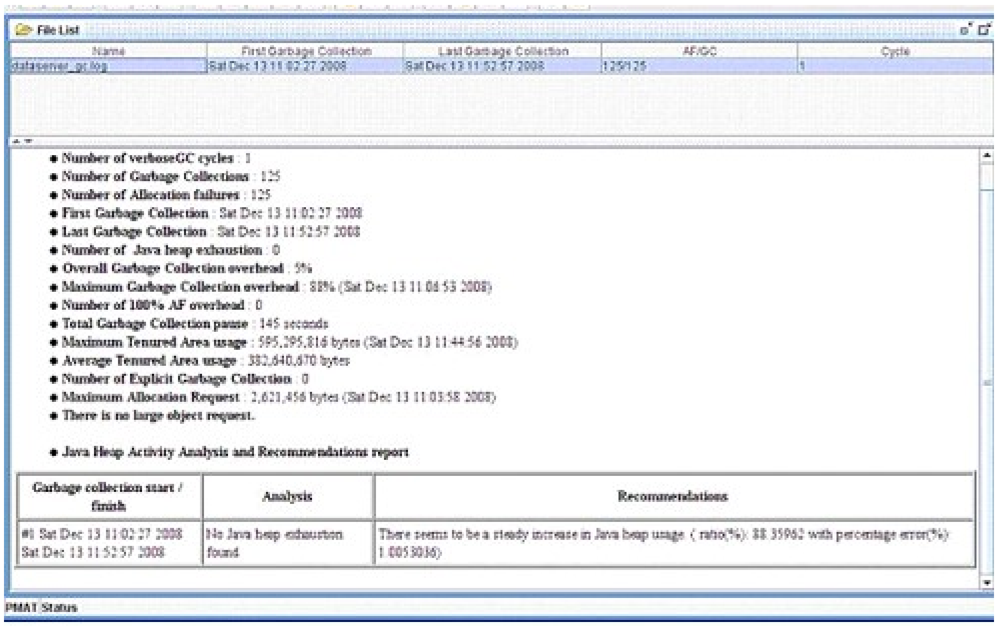
5. At this point, restart both servers, and rerun the same scenario as before. After it is complete, review the new GC logs in PMAT to determine changes in TBSM performance.

6. Review the Results after Tuning

Data Server

The following screen capture shows the result after the new garbage collection log has been opened within the PMAT tool for the TBSM Data server.

IBM Pattern Modelling and Analysis Tool for Java Garbage Collector
File Analysis View Help


File List

Name	First Garbage Collection	Last Garbage Collection	AF/GC	Cycle
dataserver_gc.log	Sat Dec 13 11 02 27 2008	Sat Dec 13 11 52 57 2008	125/125	1

- **Number of verboseGC cycles** : 1
- **Number of Garbage Collections** : 125
- **Number of Allocation failures** : 125
- **First Garbage Collection** : Sat Dec 13 11:02:27 2008
- **Last Garbage Collection** : Sat Dec 13 11:52:57 2008
- **Number of Java heap exhaustion** : 0
- **Overall Garbage Collection overhead** : 5%
- **Maximum Garbage Collection overhead** : 88% (Sat Dec 13 11:08:55 2008)
- **Number of 100% AF overhead** : 0
- **Total Garbage Collection pause** : 145 seconds
- **Maximum Tenured Area usage** : 595,295,816 bytes (Sat Dec 13 11:44:56 2008)
- **Average Tenured Area usage** : 382,640,670 bytes
- **Number of Explicit Garbage Collection** : 0
- **Maximum Allocation Request** : 2,621,456 bytes (Sat Dec 13 11:03:58 2008)
- **There is no large object request.**

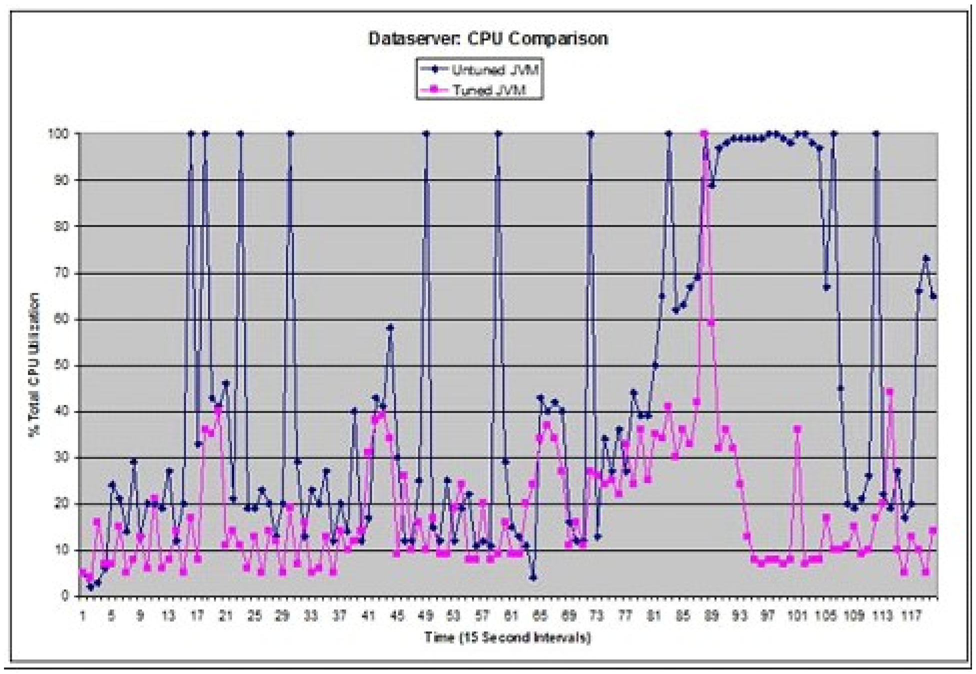
- **Java Heap Activity Analysis and Recommendations report**

Garbage collection start / finish	Analysis	Recommendations
#1 Sat Dec 13 11:02:27 2008 Sat Dec 13 11:52:57 2008	No Java heap exhaustion found	There seems to be a steady increase in Java heap usage. (ratio(%): 88.35962 with percentage error(%): 1.0053036)

PMAT Status

After the run is complete, load the new Dataserver_gc.log file into the PMAT tool and review the Analysis and Recommendations sections. For this "tuned" scenario, the analysis section again indicates that no Java heap exhaustion was found. However, Overall Garbage Collection Overhead is now calculated at 5% (down from 20% prior to tuning), representing a reduction of 15%. Less time spent in garbage collection essentially translates to more processer cycles available for application processing.

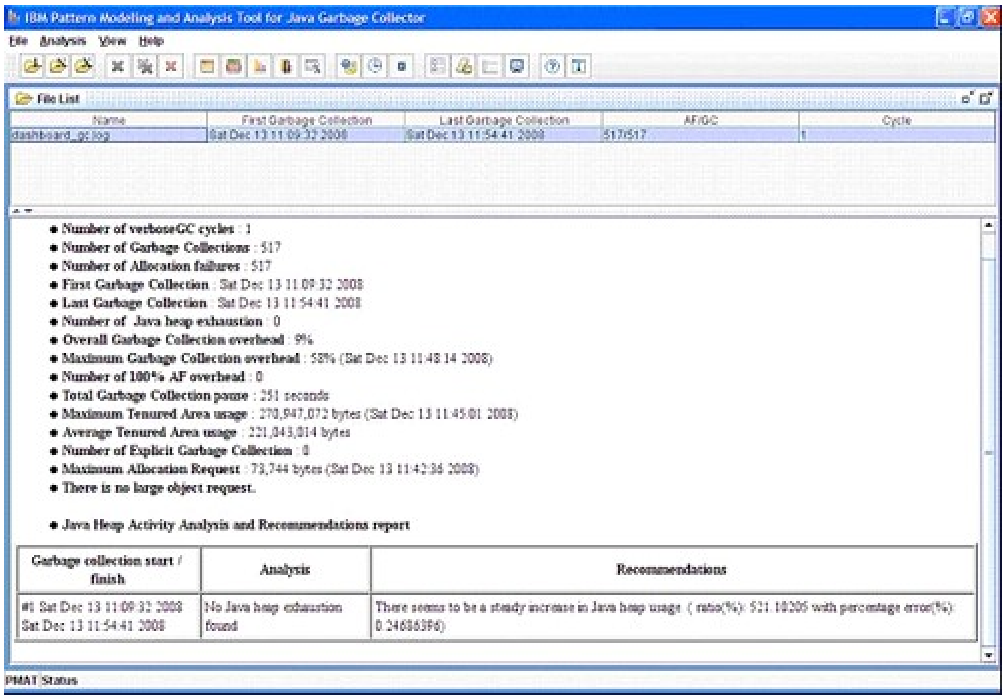
To illustrate the CPU savings for the Data Server, the vmstat data for total processor utilization was collected and plotted in the following chart for the event processing workload of 30 minutes (Note that these CPU comparison results are not a guarantee of service or performance improvement; the performance of each unique TBSM environment will vary):



For the initial event processing workload, the average processor utilization was 43.4% of total CPU on the Data server system. After tuning, the same workload used an average of 18.3% of total processor utilization, a reduction of 57.9% of processor overhead. Also, an extended period of 100% processor utilization late in the run was almost entirely eliminated in the tuned environment.

Dashboard Server

The following screen capture shows the result after the new garbage collection log has been opened within the PMAT tool for the TBSM Dashboard Server.


IBM Pattern Modeling and Analysis Tool for Java Garbage Collector
File Analysis View Help
File List
PMAT Status

Name	First Garbage Collection	Last Garbage Collection	AF/GC	Cycle
dashboard_gc.log	Sat Dec 13 11:09:32 2008	Sat Dec 13 11:54:41 2008	517/517	1

- **Number of verboseGC cycles** : 1
- **Number of Garbage Collections** : 517
- **Number of Allocation failures** : 517
- **First Garbage Collection** : Sat Dec 13 11:09:32 2008
- **Last Garbage Collection** : Sat Dec 13 11:54:41 2008
- **Number of Java heap exhaustion** : 0
- **Overall Garbage Collection overhead** : 9%
- **Maximum Garbage Collection overhead** : 58% (Sat Dec 13 11:48:14 2008)
- **Number of 100% AF overhead** : 0
- **Total Garbage Collection pause** : 251 seconds
- **Maximum Tenured Area usage** : 270,947,072 bytes (Sat Dec 13 11:45:01 2008)
- **Average Tenured Area usage** : 221,043,014 bytes
- **Number of Explicit Garbage Collection** : 0
- **Maximum Allocation Request** : 73,744 bytes (Sat Dec 13 11:42:36 2008)
- **There is no large object request.**

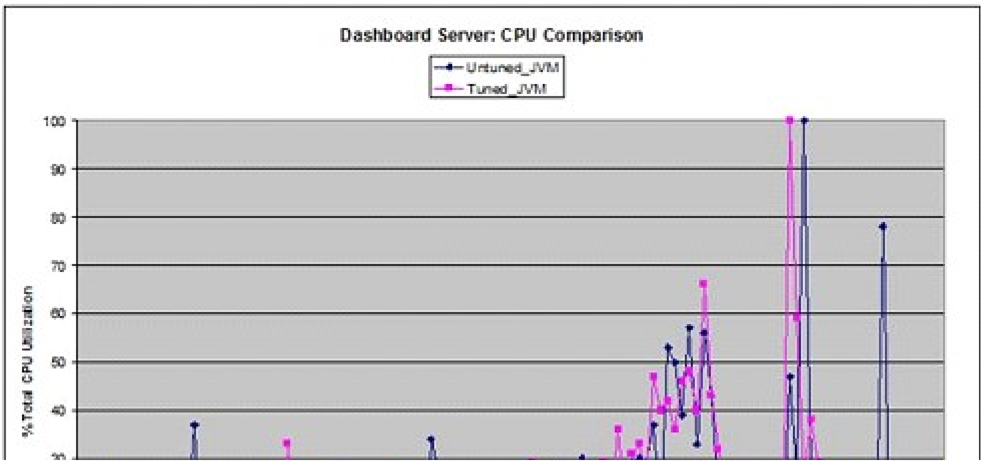
- **Java Heap Activity Analysis and Recommendations report**

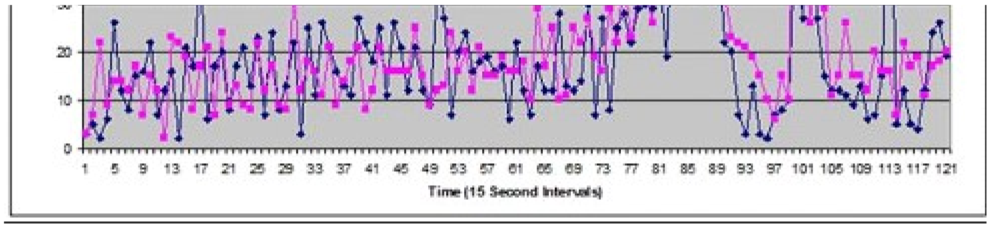
Garbage collection start / finish	Analysis	Recommendations
#1 Sat Dec 13 11:09:32 2008 Sat Dec 13 11:54:41 2008	No Java heap exhaustion found	There seems to be a steady increase in Java heap usage. (ratio(%): 521.10205 with percentage error(%): 0.24686396)

Review the Analysis and Recommendations sections. For this "tuned" scenario, the analysis section again indicates no Java heap exhaustion. Garbage Collection overhead is now calculated at 9% (down from 10% prior to tuning), which seems to be a minimal gain.

However, an interesting metric to consider is the Total Garbage Collection pause, which is now 252 seconds, down from 288 seconds in the original (untuned) Dashboard Server scenario. As previously stated, application processing is essentially paused while some garbage collection activities occur. Although each of these pauses can range from several milliseconds to several hundred milliseconds each (spread over time and unique to each environment), a reduction in the total number of overall garbage collection time is another worthwhile metric to consider.

Finally, to illustrate the CPU comparison for the Dashboard Server, the vmstat data for total processor utilization was plotted in the following chart for the same event processing workload. Again note that these CPU results are not a guarantee of service or performance improvement; performance for each unique TBSM environment will vary:





For the initial event processing workload, the average processor utilization was 19.1% of total CPU on the Dashboard Server. After tuning, the same workload now used an average of 20.2% of total processor utilization, a minimal increase in total system processor utilization.

This illustrates an important concept: the larger the maximumHeapSize value, potentially, the more objects are loaded into JVM memory because there is additional memory space. Therefore, additional CPU processing is most likely needed by the GC threads to purge the larger JVM heap of unreachable objects.

While a minor gain in overall processor utilization was discovered using the larger maximumHeapSize setting, the cost savings of 36 fewer seconds (over the 30 minute period) spent in GC pause time might or might not be worth the trade off. However, in a production environment, you might want to consider conducting a longer scenario (perhaps over the course of a day) to determine if the larger setting or smaller setting is a better choice.

This is just a basic, but powerful example of how you can use the PMAT tool to become familiar with your own environment. As TBSM environments are unique per customer, making educated tuning decisions by using the PMAT tool is a recommended strategy for performance tuning._Performance and Tuning.html

_TBSM 4.2.1 - 6.1 Performance & Tuning

These topics cover Performance and Tuning for TBSM versions 4.2.1 - 6.1. You can also download the most recent pdf version of this document:

TBSM 4.2.1 - 6.1 Performance Tuning Whitepaper

IBM® Tivoli® Business Service Manager (TBSM) delivers technology for IT and business users to visualize and assure the health and performance of critical business services. The product does this by integrating a logical representation of a business service model with status-affecting alerts that are raised against the underlying IT infrastructure. Using browser-based TBSM Web Consoles, operators can view how the enterprise is performing at a particular time, or how it performed over a given period of time. As a result of this, TBSM delivers the real-time information that you need to respond to alerts effectively and in line with business requirements, and optionally to meet Service Level Agreements (SLAs).

Given the size of today's large business enterprises, TBSM must be able to represent and manage the status and related attributes of very large business service models. To enhance scalability, TBSM 4.2 divided the previous TBSM 4.1.x server architecture into two separate servers, referred to in this paper as the "Data Server" for back-end processing, and "Dashboard Server" for front-end operations. For additional benefits, TBSM 6.1 introduced support for 64-bit operating systems to meet the needs of Customers requiring high levels of scalability in terms of sheer service model size as well as support for large numbers of concurrent Web Console operators.

For reference, the Data Server maintains the canonical TBSM business service model representation, processing events from various sources, and updating service status based on those events. In this role, it interacts with various data stores. The Dashboard Server, by contrast, is primarily responsible for supporting the user interface. It retrieves service information from the Data Server as needed to support the user interactions.
TBSM is primarily processor dependent (the number and speed of processors being two of the key factors) as long as sufficient memory is configured for the TBSM Java ™ Virtual Machines (JVMs). It is important to be aware of the minimum and recommended hardware specifications for an optimal user experience (provided later

in this document).

To that end, the purpose of this paper is to describe some of the performance tuning methodologies and configuration options available for you to use with the product, how to interpret and analyze the results of performance data collections, and to provide recommendations for installing and tuning the product to achieve optimal scale and performance in your own unique TBSM environment.

Tivoli Netcool/Impact

Tips (tbsm)

Like | Updated Monday at 9:46 AM by GERALDINE MCCORMACK | Tags: *None* Add tags

Edit Page Actions

The topics in this section provide guidance for processes and proceures not covered in the other TBSM sections.

Table of Contents:

Monitoring the Status of the TBSM XML Toolkit

This document link below is a guide to adding an ITM situation to monitor the status of the TBSM XML Toolkit within the Monitoring Agent for Business Service Manger Common Agent(TBSM Common Agent) workspace.

The doument can be found here XMLToolkitStatus.pdf

Comments (0) Versions (3) Attachments (0) About

There are no comments.

Add a comment

Feed for this page | Feed for these comments

w3-connections.ibm.com

Migration (tbsm) - Tivoli Netcool/Impact Wiki

16-20 minutes

The topics in this section provide guidance on deploying the following products that are deployed in Tivoli Integrated Portal with a focus on their cross-product migration and export/import capabilities and considerations:

This topic describes general information you should consider when you migrate into an environment with multiple products.

The vault key is an encryption key that is used to encrypt the administrator password that was provided during installation and is stored locally for Tivoli Integrated Portal applications.

There is a known issue exporting the Web GUI 7.3.1 component from an ITNM 3.9 system on Tivoli Integrated Portal 2.1 and importing into ITNM 3.9 on Tivoli Integrated Portal 2.1 or Tivoli Integrated Portal 2.2.

- The vault.key file is copied when it should not be, and this causes the Web GUI password validation to fail.
- Fixed in Web GUI 7.3.1 FP1, which must be applied to export system.

Workarounds if Web GUI FP1 is not installed:

- Pre-emptive on export system:

  Edit the file *omnibus_webgui*/integration/plugins/OMNIbuswebGUI_clone_settings.properties to ignore the EXPORT of etc/encrypt/vault.key.

- Pre-emptive on target system:

  Edit the file *omnibus_webgui* /integration/plugins/OMNIbuswebGUI_clone_settings.properties to ignore the IMPORT of etc/encrypt/vault.key (1 line to edit).

  If pre-emptive action not taken:

1. Edit the file *$NCHOME*/omnibus_webgui/etc/datasources/ncwDataSourcesDefinitions.xml file, as follows:

1. Locate the following entry:

   <ncwDataSourceCredentials userName="root" password="vA4UMSGAMGpo3mIE4DjlUw==" encrypted="true" algorithm="FIPS" />

2. Edit the entry to read as follows:

   <ncwDataSourceCredentials userName="root" password="" encrypted="false" />

2. You can use the imported vault.key to generate a new encrypted password, see the Web GUI administrator guide for information on how to generate encrypted passwords.

   "ESS plugin with missing jar" errors with the upgrade framework:

- Web GUI workaround to ignore the ESS plugin:

Prior to running preupgrade/upgrade scripts carry out the following changes to avoid the ESS server error:

- Preupgrade:

  Edit *tip_home_dir*/profiles/TIPProfile/upgrade/plugins/OMNIbusWebGUI.properties to comment out the following line:

  components=ESSServer

- Upgrade:

  Edit *$NCHOME*/omnibus_webgui/integration/plugins/OMNIbusWebGUI.properties to comment out the following line:

  components=ESSServer

Chart data when migrating Network Manager and TBSM

When migrating on a system with Network Manager and TBSM installed, the Tivoli Integrated Portal import function sets all chart connection information, regardless whether it is run by Network Manager or TBSM. Given that Network Manager does check for this, running the Network Manager import operation corrupts TBSM chart connection information in that the EventSummary panel fails to connect to the datasource.

The TBSM Dashboard server upgrade script calls the tipcli ChartConnection --action create command to re-create (effectively update) the TBSMChartService connection information to have the correct port for the TBSM 6.1 environment. This only happens when TBSM is upgraded, but the problem is happening during the upgrade of Network Manager.

The Tivoli Integrated Portal charting migration plugin always exports all chart connection information, as this is not tied to any particular application. Thus, when Network Manager is upgraded, the old connection information, including the old port number, is imported into the TBSM 6.1 derby database.

Thus, the problem described here will occur in the following scenario:

1. A preupgrade is run on a Tivoli Integrated Portal 1.1 system that includes TBSM 4.2 or 4.2.1.
2. TBSM 6.1 has already been installed on the target and the exported data from step 1 is imported as part of an upgrade command that is not for TBSM.
3. The TBSM upgrade command had been run previously, or there are no plans to run the TBSM upgrade command.

The simplest solution, where appropriate, is to make sure the TBSM upgrade command is run last. When this is not practical, the connection can be repaired with the following command:

tipcli ChartConnection --action create --name TBSMChartService --protocol http --hostname localhost --port 16310 --serviceName ibm/sla/rad --renderFormat BIRT --credentialType SSO --username *tbsm_tip_admin_user* --password *tbsm_tip_admin_password*

where:

tbsm_tip_admin_user and *tbsm_tip_admin_password* must be set to the correct values for the Dashboard server administrative user. The value 16310 is the default port, but the TBSM command that runs during upgrade gets the port number from property WC_defaulthost found in *tip_home_dir*/properties/TIPPortDef.properties

To migrate Tivoli Common Reporting reports, refer to the Tivoli Common Reporting documentation. Tivoli Common Reporting 2.1, and higher, does not migrate JNDI datasources. TBSM 4.2.1 reports use JNDI datasources. They do not execute after migration unless you have updated the datasource to use JDBC.

TBSM 6.1 provides an installation script to load and configure a set of Cognos reports and the legacy TBSM 4.2.1 BIRT reports. It configures the BIRT reports with a JDBC datasource. If you have modified the TBSM 4.2.1 reports and want to migrate them, or if ITMN migration has migrated the TBSM 4.2.1 reports, then you must update the BIRT report datasource. The names of the BIRT reports loaded by TBSM 6.1 have been qualified by the release, so there are no naming conflicts between the TBSM 4.2.1 and TBSM 6.1.

To update the JNDI datasource to JDBC, use the following steps:

1. Make a copy of the following file:

   tip_home_dir/tipv2Components/TCRComponent/data/resource /tbsm/resources/tbsm/lib/tbsm.rptlibrary

2. In a text editor, edit tbsm.rptlibrary, as follows:

1. Find the datasource definition:

   <data-sources>

   <oda-data-source extensionID="org.eclipse.birt.report.data.oda.jdbc" name="ITM Warehouse" id="4">

   <property name="odaJndiName">jdbc/ibm/tivoli/tbsm/v4</property>

   </oda-data-source> <script-data-source name="ReportPeriodChoicesDS" id="1620"/>

   </data-sources>

2. Replace the datasource definition with the following:

   <data-sources>

   <oda-data-source extensionID="org.eclipse.birt.report.data.oda.jdbc" name="ITM Warehouse" id="4">

   <property name="odaDriverClass">com.ibm.db2.jcc.DB2Driver</property>

   <property name="odaURL">jdbc:db2://DB2_HOST_NAME:DB2_PORT /DB2_DATABASE_NAME:currentSchema=DB2_SCHEMA;</property>

   <property name="odaUser">DB2_USERID</property>

   <encrypted-property name="odaPassword" encryptionID="base64">SVRNVVNFUg==</encrypted-property>

   </oda-data-source>

   <script-data-source name="ReportPeriodChoicesDS" id="1620"/>

   </data-sources>

3. After migrating the TBSM 4.2.1 BIRT reports, invoke the Tivoli Common Reporting command (*tip_home_dir*/tipv2Components/TCRComponent/bin/trcmd) to update the datasource definition (with information about your IBM Tivoli Data Warehouse) and apply it to the migrated TBSM 4.2.1 reports.

   trcmd -modify

   -datasources

   -reports

   -reportname

   "/content/package[@name='Tivoli Products']/folder[@name='Tivoli Business Service Manager']/folder[@name='Tivoli Business Service Manager

Summary']/report[@name='tbsm_availability_summary_table']"

-username *tip_user_id*

-password *tip_user_password*

-setdatasource

odaDriverClass=*jdbc_driver_class*

“odaURL=*jdbc_URL*"

odaUser=*database_userid*

odaPassword=*database_password*

The *jdbc_URL* specifications are based on the database type shown below. The drivers should be copied to the following directory:

*tip_home_dir/*tipv2Components/TCRComponent/lib/birt-runtime-2_2_2/ReportEngine/plugins /org.eclipse.birt.report.data.oda.jdbc_2.2.2.r22x_v20071206/drivers.

If you install the TBSM 6.1 reports (or have already done so), the driver files are copied to the directory by the installation script.

- DB2
- Driver class: com.ibm.db2.jcc.DB2Driver
- Jar files: db2jcc.jar, db2jcc_license_cu.jar
- JDBC URL: jdbc:db2://*hostname*:*port*/*database*:currentSchema=ITMUSER;
- Oracle
- Driver class: oracle.jdbc.driver.OracleDriver
- Jar file: ojdbc6_g.jar
- JDBC URL: jdbc:oracle:thin:@*hostname*:*port*:*database*
- MS SQL
- Driver class: com.microsoft.sqlserver.jdbc.SQLServerDriver
- Jar file: sqljdbc.jar
- JDBC URL: jdbc:sqlserver://*hostname*:*port*;databasename=*database*:currentSchema=ITMUSER;

4. Run a report.

   Note: You do not need to restart TBSM.

Preupgrade.tar when migrating multiple products

When migrating a system with more than one product installed, for example, with both Network Manager and TBSM installed, you must delete or rename the *tip_home_dir*/profiles/TIPProfile/upgrade directory on the export/source system prior to extracting a product’s Preupgrade.tar file.

This should be done for each product in turn.

Essentially, do not extract multiple Preupgrade.tar files into a single upgrade directory and share that directory.

This topic describes issues you may encounter when you migrate in environments with specific Tivoli products.

Deployment Engine issue

If there is a pre-existing product using Deployment Engine (DE) 1.4.0.13 or higher, you cannot install Network Manager and OMNIbus ObjectServer using the Network Manager installation package.

To determine the DE version, at the command line run the following command:

DE_home/bin/de_version.cmd
where *DE_home* is the installation location for the Deployment Engine, for example, on Windows systems the default installation location is C:\Program Files\IBM\Common\acsi.

As a workaround, install Tivoli Netcool/OMNIbus first using the OMNIbus installation package and then install Network Manager using its installation package.

Tivoli Common Reporting 2.1.1 is a prerequisite

You cannot install Network Manager 3.9 after TBSM 6.1 (or Tivoli Integrated Portal 2.2) without first installing Tivoli Common Reporting 2.1.1.

In this situation, you must install Tivoli Common Reporting 2.1.1, before Network Manager 3.9, otherwise, Network Manager 3.9 will attempt to install Tivoli Common Reporting 2.1, which will fail in the Tivoli Integrated Portal 2.2 environment.

NOTE: As of 6.1.1.4 TBSM and Netcool Impact no longer support TCR 2.*. TCR 3.1+ must be installed with JazzSM server.

Installing Network Manager after TBSM 6.1

If you are installing Network Manager 3.9 after TBSM 6.1 with Tivoli Integrated Portal 2.2.0.3, you must keep the Tivoli Netcool/OMNIbus ObjectServer running.

Tivoli Integrated Portal portlets render incorrectly

If portlets in the Tivoli Integrated Portal Server, such as in the Active Event List or topology views, render as black in color, you may need to configure a workaround, as follows:

Environment: All client/end-user browsers and operating systems; Oracle JRE 1.6.0_21 to 1.6.0_23 or IBM JRE 1.6.0 SR9.

Symptom:** **When attempting to use any of the Network Manager IP applet-based views (for example, Network View, Hop View) for the first time, or after the applet cache has been cleared from the Java Control Panel, the applets will not download, the applicable screen will freeze or display as a black panel, and, if enabled, the Java Console will crash.**

Cause: **A change to the JRE checking algorithms caused a hardcoded heartbeat timeout to be exceeded, and the JRE to terminate ungracefully, during the download of applets where (1) the applet jar files are large (2) the network is contended (3) or both. Note that as this is in part down to network conditions, the occurrence of this problem is intermittent.**

Solution:** On the client/end-user machine set the following environment variable:

JPI_PLUGIN2_NO_HEARTBEAT = 1 and restart the browser.

IBM has made an enhancement request to Oracle that this variable be parameterized, so that in future it can be passed to the JRE by the applet at launch, and obviate the need to set it on a per end-user basis.

Exporting Network Manager GUI data fails with incorrect file permissions (Linux and UNIX only)

Problem (Abstract)

On Linux and UNIX operating systems, the Network Manager GUI export utility (nmGuiExport) might fail due to permission problems. This only occurs when TBSM 6.1 is installed on a server that already has a Network Manager 3.9 installation.

Resolving the problem

Make sure the file permissions are set to 755 for the $TIPHOME/profiles/TIPProfile/upgrade/bin/preupgrade.sh file, then run the Network Manager GUI export utility (nmGuiExport) again.

IBM Java Runtime Environment Version 1.6

Web GUI 7.3 supports only version 1.6 of the IBM supplied Java Runtime Environment (IBM JRE 1.6).

Prerequisite when installing Web GUI 7.3 over Network Manager 3.9

Before installing Web GUI 7.3 into an environment that includes Network Manager 3.9, you must edit the following file to change its WebSphere Application Server setting from Base to eWAS:*tip_home_dir*/profiles
/TIPProfile/properties
/.tipinfo.properties

The Tivoli Integrated Portal feature pack upgrades Tivoli Integrated Portal 2.1.0.*x* to Tivoli Integrated Portal 2.2.0.1 and it may be a pre-requisite step prior to installing another product into the same Tivoli Integrated Portal container.

Considerations:

Make sure you stop ALL java processes prior to running the upgrade.

Make sure your Tivoli Integrated Portal user ID and password are correct in the silent response file. Otherwise, the upgrade will run for some time and then fail.

Tivoli Integrated Portal can be installed into an existing base WebSphere Application Server (WAS), also known as a full WAS. Other Tivoli Integrated Portal based products may or may not support this feature.

Use of a custom context root is not supported

Tivoli Integrated Portal supports changing the context root. Web GUI supports this but TBSM does not currently support it and will not work if it has been changed. The TBSM installer blocks installing into a Tivoli Integrated Portal where the context root is not /ibm/console. The same is true for Tivoli Netcool/Impact 6.1.

Installation into a WAS base is not supported

TBSM does not support installation into an existing base WebSphere Application Server (WAS) also known as full WAS:

Tivoli Integrated Portal can be installed into base WAS. Other Tivoli Integrated Portal based products may support this. TBSM does not currently support installing into an existing base WAS. The TBSM installer blocks installation into an existing Tivoli Integrated Portal instance that has been installed into base WAS. The same is true for Impact 6.1.

Install Tivoli Integrated Portal 2.2.0.1 feature pack before TBSM 6.1

The Tivoli Integrated Portal 2.2.0.1 feature pack upgrade must be run prior to installing TBSM 6.1 over products running Tivoli Integrated Portal 2.1.

Update ObjectServer schema before installing TBSM

If you are installing TBSM with a pre-existing Tivoli Netcool/OMNIbus ObjectServer, you must update the ObjectServer schema before installing TBSM. Otherwise the install will run for some time and then fail. Follow these steps to update the schema.

At the command line, change to directory to:

TBSM_install_image/TBSM/omnibus/schema_files
Type

tbsm_db_update.sql | \IBM\tivoli\netcool\omnibus\bin\isql.bat –S NCOMS –U root –P *password*
(Web GUI FP2 only – new subsection) Locate and update the following properties file

tip_home_dir\TIPProfiles\properties\.tipinfo.properties to change BASE to eWAS.
Also do this when installing Web GUI FP2 over Network Manager 3.9.

General considerations

***** NOTE**: As of 6.1.1.4 TBSM and Netcool Impact no longer support TCR 2.*. TCR 3.1+ must be installed with JazzSM server. Installing TCR into an existing TIP is no longer supported!

When installing into an existing Tivoli Integrated Portal, make sure you select the proper option for reuse rather than “install new instance of Tivoli Common Reporting”.

Restart Tivoli Common Reporting after all other products are up and running:

/opt/IBM/Tivoli/tipv2Components/TCRComponent/bin
/stopTCRserver.sh / startTCRserver.sh

Tivoli Integrated Portal version

It should be noted that the Tivoli Common Reporting 2.1.1 may be running in a Tivoli Integrated Portal 2.1 or 2.2 environment:

When you install Tivoli Common Reporting 2.1.1, it also installs Tivoli Integrated Portal 2.2.

When you install Tivoli Common Reporting 2.1.1 over Tivoli Common Reporting 2.1, the underlying environment remains as Tivoli Integrated Portal 2.1.