Solution Guide for Integrating IBM Tivoli System Automation Application Manager with IBM Tivoli Business Service Manager

August 23, 2010

IBM

Author
Wolfgang Schäberle
wschaebe@de.ibm.com

Abstract
This document presents a solution showing the integration of IBM® Tivoli® Business Service Manager (TBSM) and IBM Tivoli System Automation Application Manager to monitor, control and automate your business applications from an end-to-end point of view.

Introduction

Typically, business applications consist of different middleware components, are multi-tiered, and run on heterogeneous platforms. IBM® Tivoli® Business Service Manager (TBSM) is used to provide health 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 application landscape, 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 to automate start and stop dependencies, provide a common operating environment, provide automatic recovery in failure situations, and to get an aggregated availability status.

IBM Tivoli System Automation for Multiplatforms and IBM Tivoli System Automation for z/OS are used to make individual components of the business application highly available, for instance, a critical database.

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, and automate 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 to create the integration solution.

For more information on Tivoli Business Service Manager, see http://www.ibm.com/software/tivoli/products/bus-srv-mgr/.

For more information on Tivoli System Automation Application Manager, see http://www.ibm.com/software/tivoli/products/sys-auto-app-mgr/.

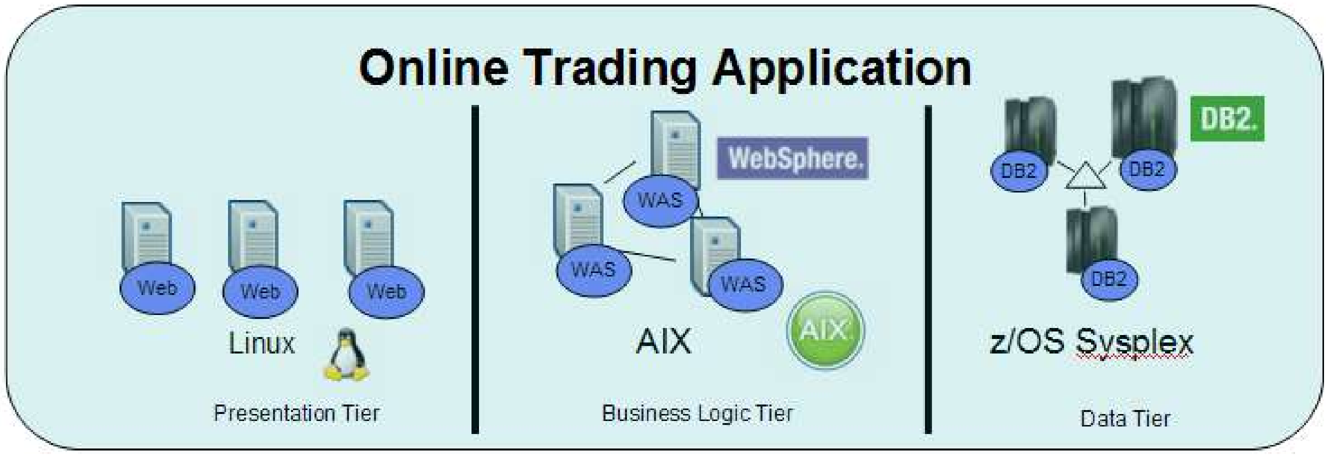
Contents

1 Part 1: Integration overview and usage scenarios

The integration solution of Tivoli System Automation Application Manager and TBSM addresses the following areas:

- Visualize service scorecards, key performance indicators, and service level agreements of business applications in TBSM. Use Tivoli System Automation Application Manager to control and automate these business applications, and to provide automatic recovery of failure situations.
- Data from System Automation Events can be used to enrich TBSM service views with information from System Automation. System Automation Application Manager delivers a TBSM service template containing preconfigured rules about how to map states from System Automation to TBSM service instances.
    - As an example, this integration allows that when a business application was brought offline due to maintenance reasons, TBSM will now be aware that this is a planned downtime of the business application and not an unplanned failure situation.
- The integration of System Automation Application Manager resources with TBSM can be automated and simplified using the Discovery Library Adaptor (DLA). By using this approach, System Automation Application Manager resources are automatically imported into a TBSM service model, the correct TBSM status rules are automatically assigned to those services, and System Automation Application Manager events are automatically matched with the correct TBSM service instances without manual configuration.
- Launch-in-context from a TBSM service view to the SA Operations Console and vice versa allows an operator to easily see the information about a specific application from both products. For example, with this capability, TBSM can be used to monitor the health of a business application and an operator can use the launch-in-context capability to jump to the SA Operations Console to start or stop this business application.

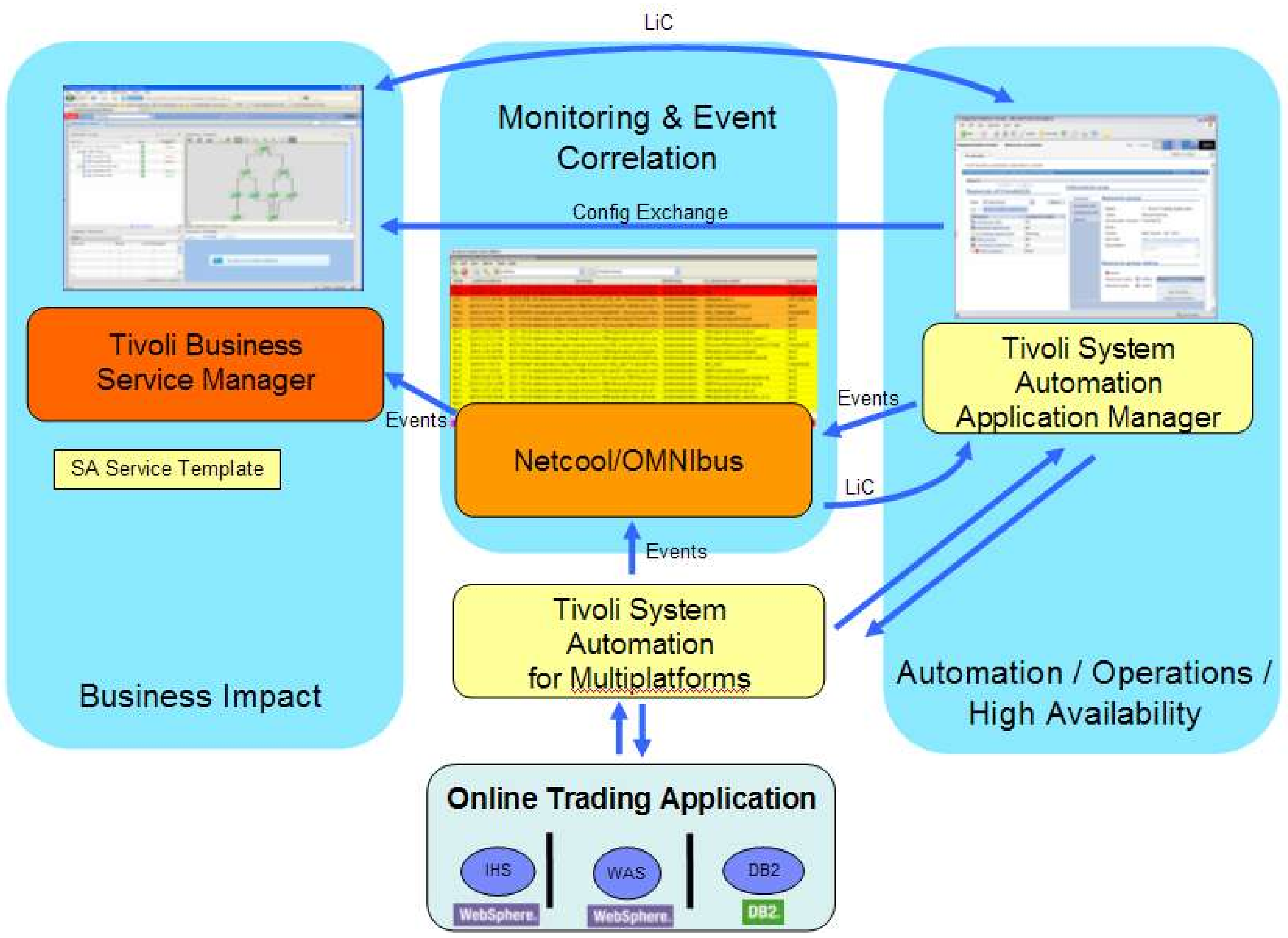
The following figure shows a typical multi-tiered, heterogeneous business application representing an "Online Trading Application":



The online trading application consists of one or more web servers in the presentation tier, an IBM WebSphere® Application Server in the business logic tier, and an IBM DB2® server in the data tier, which is made highly available. The different components run on different platforms and operating systems, and have dependencies between each other. For example, during startup, the DB2 server must first be started, then the WebSphere Application Server can be started, and finally the web servers can be started. Another dependency might be that if DB2 must be restarted, you must also restart WebSphere Application Server and the web servers for the application to resynchronize with DB2.

In these integration scenarios, the online trading application is automated and made highly available by using Tivoli System Automation and is monitored by using Tivoli Business Service Manager.

The following diagram illustrates how the key products are used in this solution:


LiC
Monitoring & Event Correlation
Config Exchange
Tivoli Business Service Manager
Events
SA Service Template
Netcool/OMNIbus
Events
Tivoli System Automation Application Manager
LiC
Events
Tivoli System Automation for Multiplatforms
Business Impact
Automation / Operations / High Availability
Online Trading Application
IHS
WAS
DB2
WebSphere
WebSphere
DB2

Tivoli System Automation for Multiplatforms

The Tivoli System Automation for Multiplatforms product monitors the application components, provides high-availability and automation of the components, and provides an availability state showing whether the application components are online or offline.

Tivoli System Automation Application Manager

The Tivoli System Automation Application Manager product manages the dependencies between the components across platform borders, provides end-to-end automated operating of the whole business application, provides recovery in failure situations, and has the knowledge about an aggregated availability state of the whole business application.

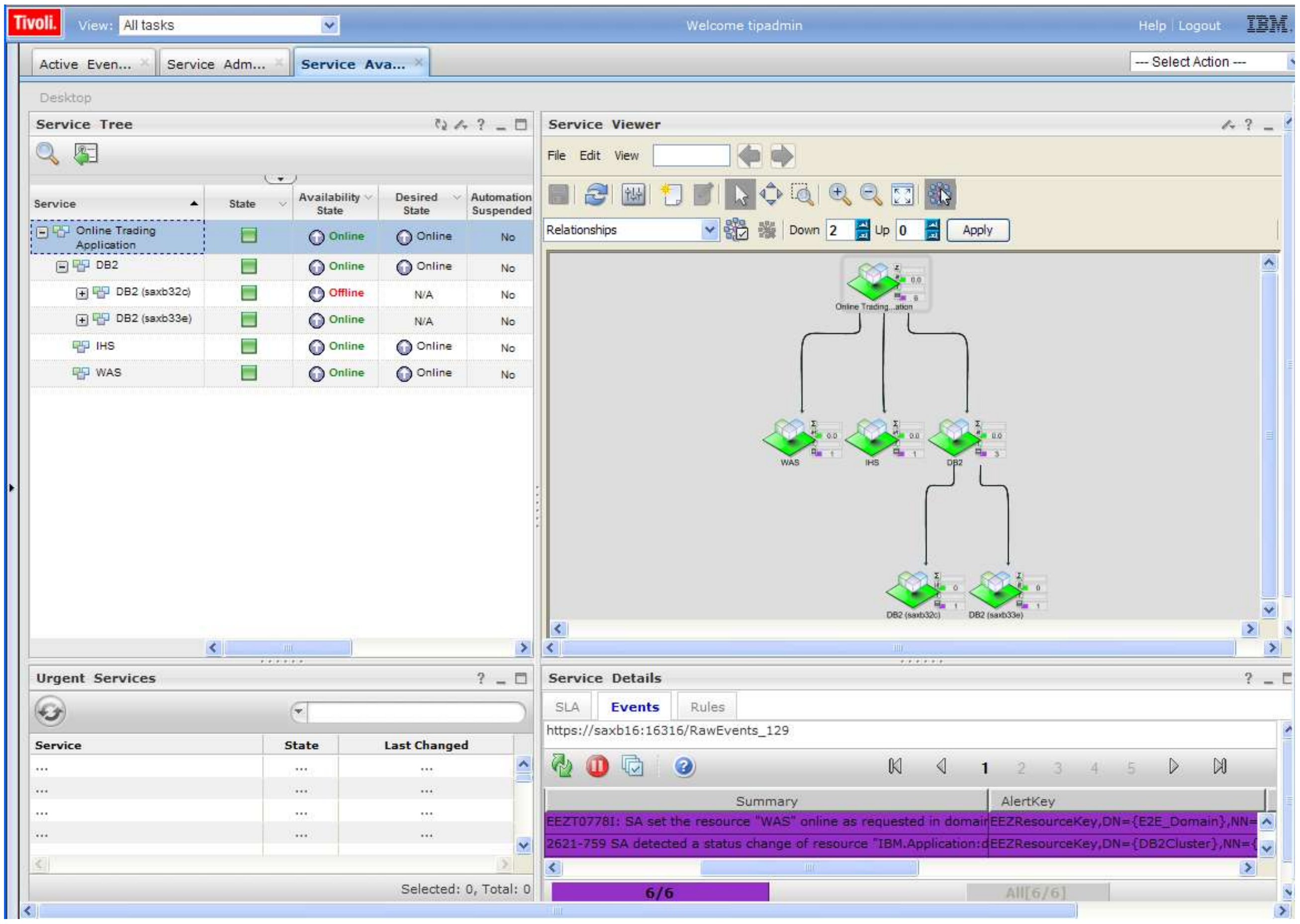
Tivoli Netcool/OMNIbus

The Tivoli Netcool/OMNIbus product is used to collect all events that are related to the business application landscape. A rules file provided by System Automation processes the events and prepares them for use in TBSM.

Tivoli Business Service Manager

TBSM visualizes the business application as a service tree. This product uses the events from Tivoli Netcool/OMNIbus to provide health information about the business application and to provide business impact analysis.

The following screen capture shows the TBSM Service Availability Page, which shows the "online trading application that is managed and monitored by using Tivoli System Automation Application Manager:



Service	State	Availability State	Desired State	Automation Suspended
Online Trading Application	🟩	Online	Online	No
DB2	🟩	Online	Online	No
DB2 (saxb32c)	🟩	Offline	N/A	No
DB2 (saxb33e)	🟩	Online	N/A	No
IHS	🟩	Online	Online	No
WAS	🟩	Online	Online	No

Service	State	Last Changed
...	...	...
...	...	...
...	...	...
...	...	...

Selected: 0, Total: 0

https://saxb16:16316/RawEvents_129

Summary	AlertKey
EEZT0778I: SA set the resource "WAS" online as requested in domain	EEZResourceKey,DN={E2E_Domain},NN=
2621-759 SA detected a status change of resource "IBM.Application:d	EEZResourceKey,DN={DB2Cluster},NN={

6/6 All[6/6]

The Service Tree shows at the top-level the "Online Trading Application" business application. The DB2, IHS (IBM HTTP Server) and WebSphere Application Server components are shown as children of the application. The DB2 component is made highly available in a High Availability Cluster managed by System Automation for Multiplatforms. Therefore, the DB2 service has two children that include the two DB2 instances in the HA Cluster: one that is currently online and the active DB2 instance, and one which is currently offline, which is the backup DB2 instance.

The Service Tree shows four status columns that present information coming from System Automation:

- **State:**
  The overall state of the service, which can be Bad ("red"), Good ("green"), or Marginal ("yellow"). This information is derived from the SA Compound State of the corresponding SA resource.
- **Availability State:**
  The currently observed run state of the service, which shows whether the service is currently online, offline, starting, or stopping.

- **Desired State:**
  Represents the automation goal of the service, which can be either Online or Offline. This state is the status in which Tivoli System Automation will try to keep the service. If it matches the Availability State, the overall state will be green.
- **Automation Suspended**
  Shows whether the automation has been suspended for the service. If the automation is suspended, Tivoli System Automation will temporarily no longer try to reach the Desired State, but will still continue to monitor the Availability State.

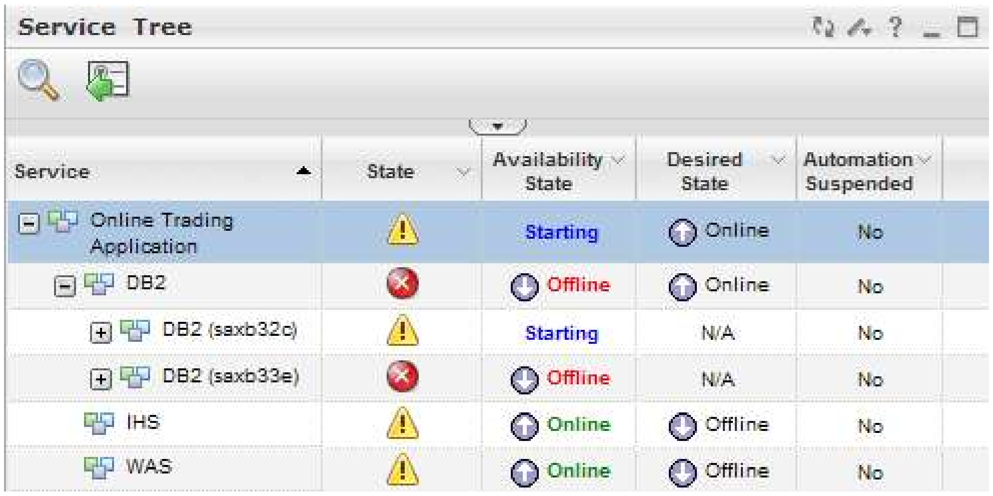
Notice in the preceding screen capture that the overall state for the DB2 backup instance is “green," although the service is not running. The backup instance shows green because it is a cold standby that is started only if the primary DB2 instance has a failure. The backup instance is intentionally not running and the operator is not required to do anything. This knowledge of a planned versus an unplanned offline state is a capability coming from System Automation events, which is reflected in the TBSM overall state of the service.

The following two usage scenarios include more details about this integration solution.

1.1 Scenario 1: Automatic recovery of a DB2 failure

Assume that the currently active DB2 instance has a failure. Tivoli System Automation detects this and tries to restart the DB2 instance in place. If this restart does not succeed, it fails over the DB2 instance to the other node in the HA Cluster.

You can monitor this situation in the TBSM Service Tree:

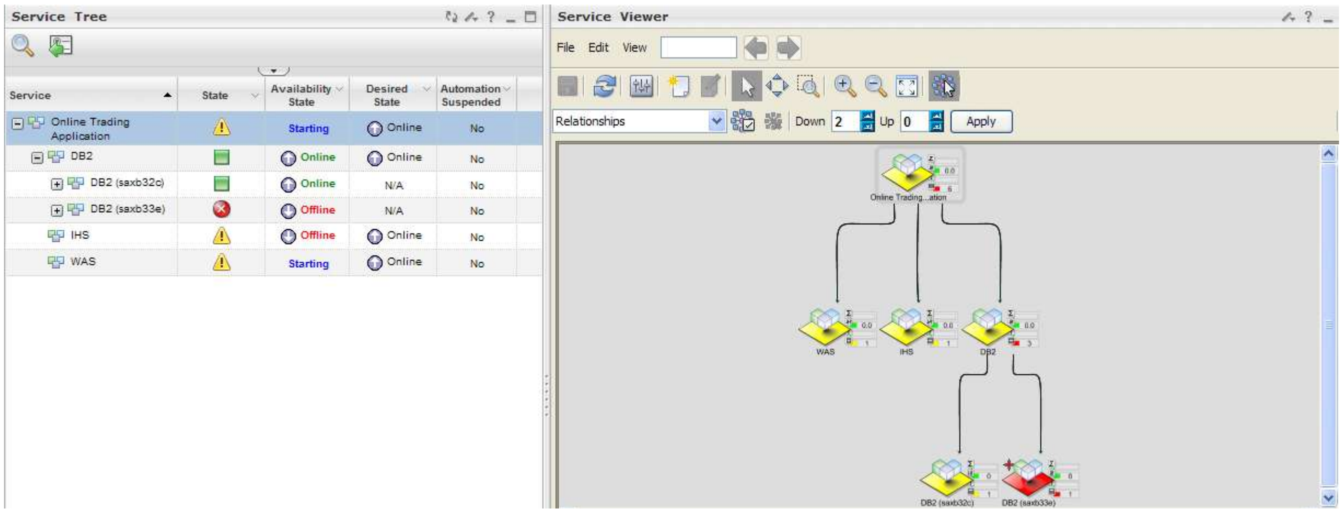


Service	State	Availability State	Desired State	Automation Suspended
Online Trading Application	⚠	Starting	Online	No
DB2	✖	Offline	Online	No
DB2 (saxb32c)	⚠	Starting	N/A	No
DB2 (saxb33e)	✖	Offline	N/A	No
IHS	⚠	Online	Offline	No
WAS	⚠	Online	Offline	No

Note that the DB2 instance that previously was online on node saxb33e now shows a problem state and is Offline. The other DB2 instance on node saxb32c is currently being started by Tivoli System Automation.

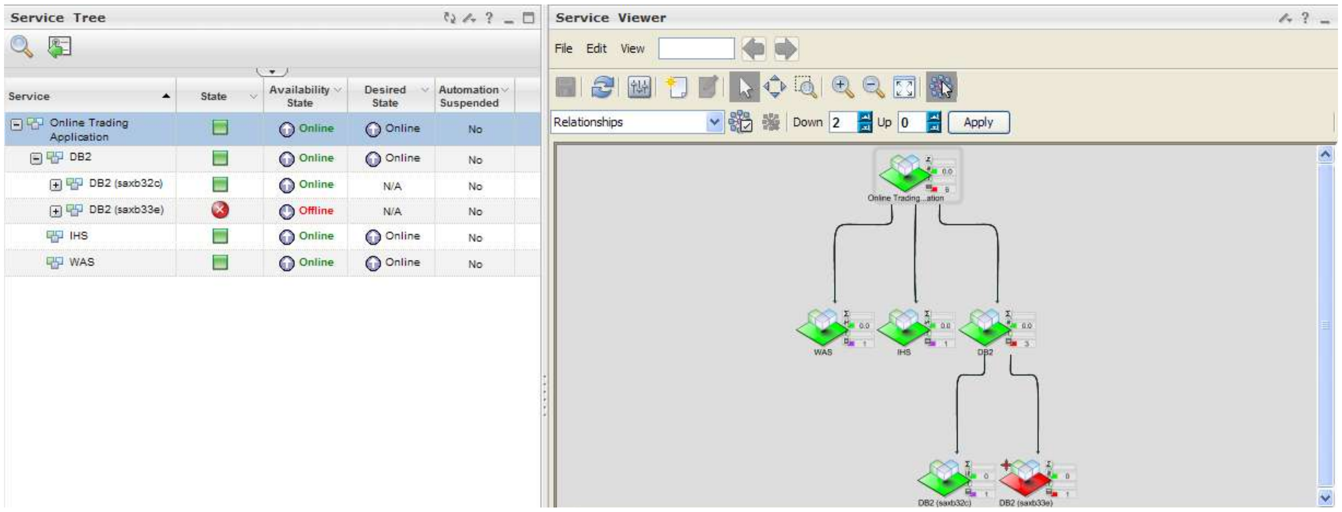
Also note that due to the failover of DB2, Tivoli System Automation Application Manager also restarts IHS and WebSphere Application Server because there is a so-called "forcedDownBy" dependency defined in the Tivoli System Automation Application Manager policy. You can see that in the following screenshot:

IHS has been stopped already and is now waiting for WebSphere Application Server to restart. DB2 has been brought online on the other node saxb3 2c in the HA Cluster.



Service	State	Availability State	Desired State	Automation Suspended
Online Trading Application	⚠	Starting	Online	No
DB2	🟩	Online	Online	No
DB2 (saxb32c)	🟩	Online	N/A	No
DB2 (saxb33e)	❌	Offline	N/A	No
IHS	⚠	Offline	Online	No
WAS	⚠	Starting	Online	No

Finally, everything is up and running again:



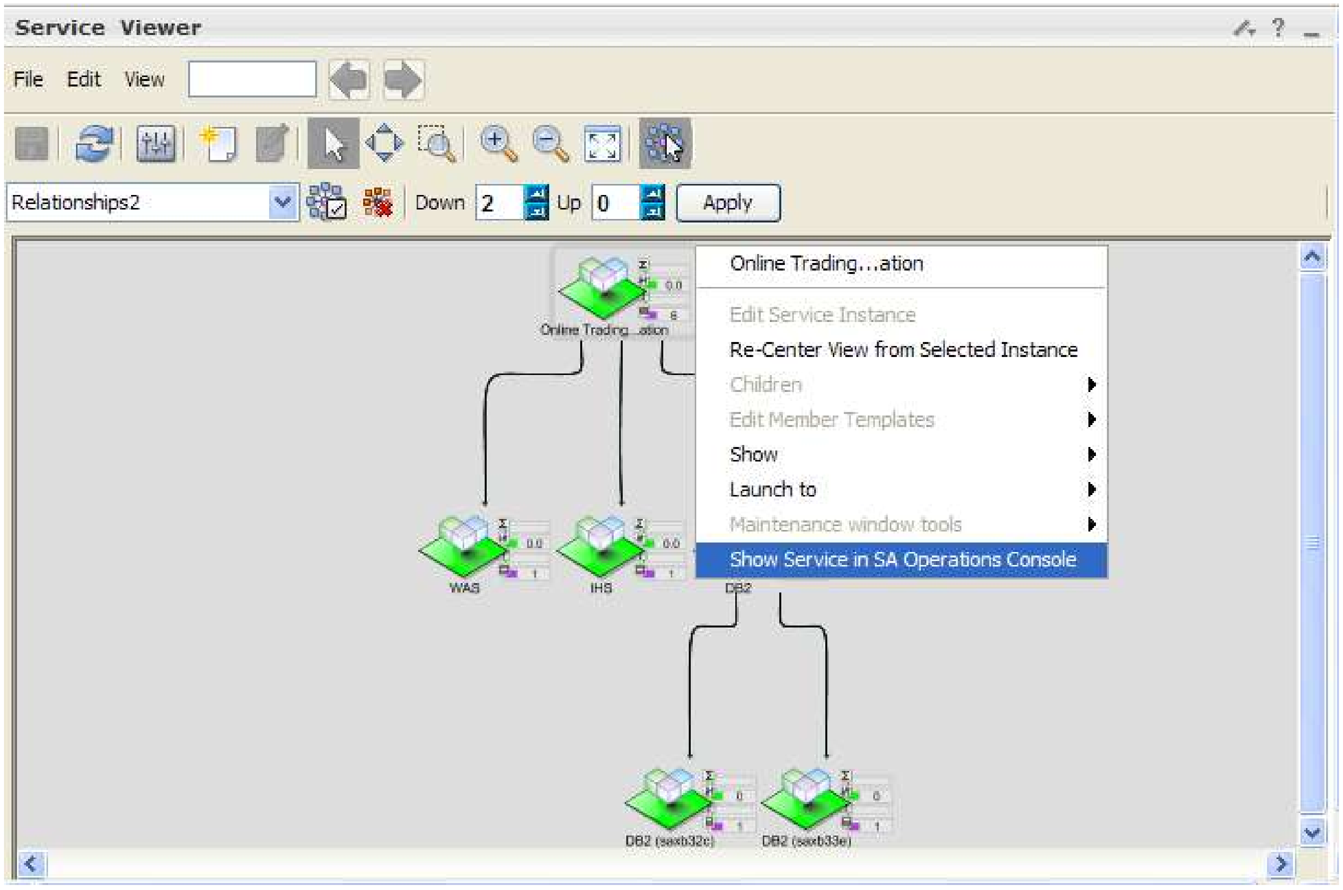
Service	State	Availability State	Desired State	Automation Suspended
Online Trading Application	🟩	Online	Online	No
DB2	🟩	Online	Online	No
DB2 (saxb32c)	🟩	Online	N/A	No
DB2 (saxb33e)	❌	Offline	N/A	No
IHS	🟩	Online	Online	No
WAS	🟩	Online	Online	No

Tivoli System Automation has automatically recovered the failure situation of DB2 by performing a failover of DB2 and by restarting the dependent WebSphere Application Server and IHS. Now, since the business application is up and running again, an administrator can take care of the failed DB2 instance and fix the problem on node saxb33e, so that the DB2 is high available again.

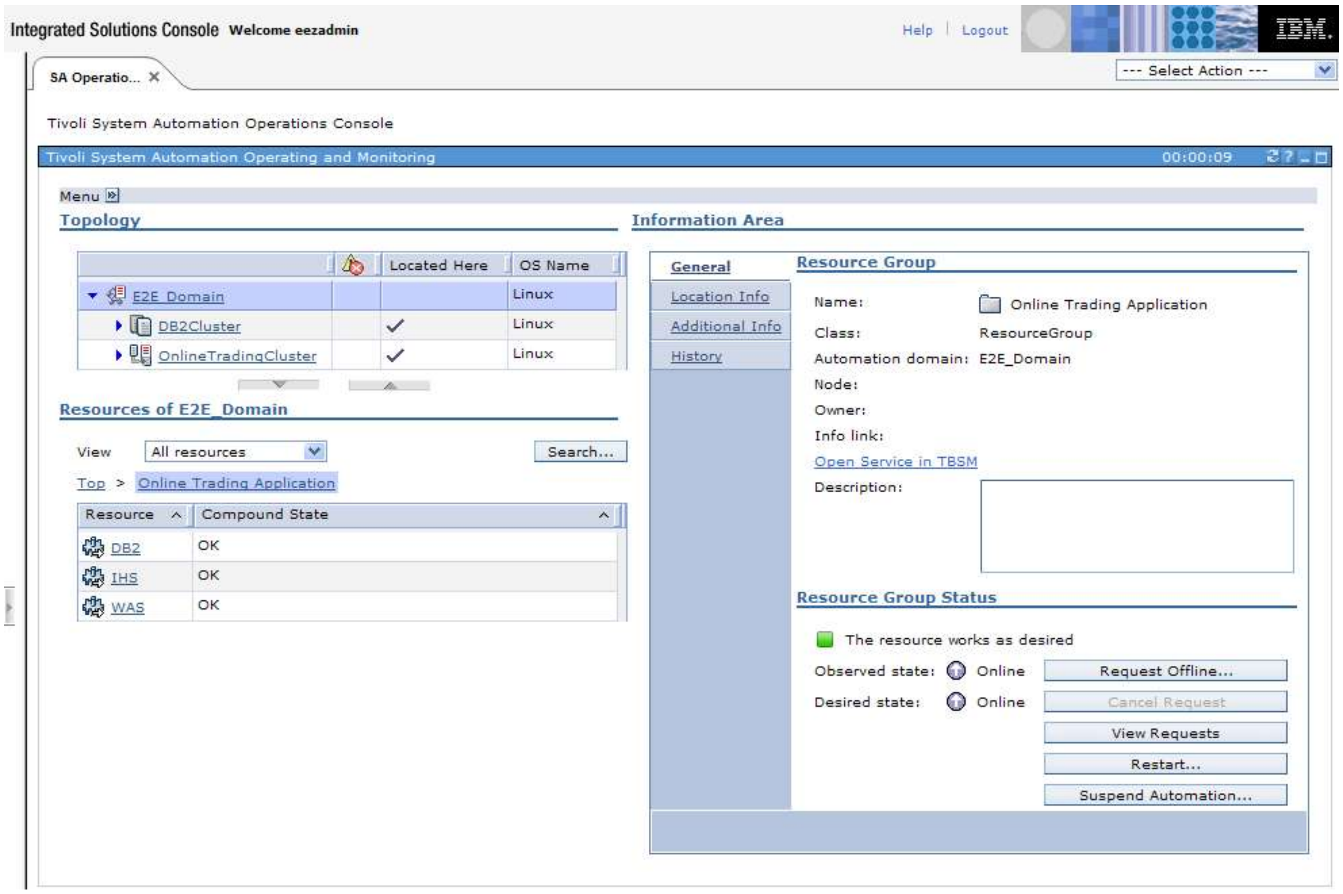
1.2 Scenario 2: Planned maintenance of "Online Trading Application"

In this scenario an operator needs to shut down the “Online Trading Application" for maintenance reasons. The operator can do this starting from the TBSM GUI by performing the following steps:

In the TBSM Service Viewer, open the context menu of the "Online Trading Application" service and select the menu entry "Show Service in SA Operations Console":



This will launch the System Automation Operations Console in the Integrated Solutions Console:



	Located Here	OS Name
E2E_Domain		Linux
DB2Cluster	✓	Linux
OnlineTradingCluster	✓	Linux

Resource	Compound State
DB2	OK
IHS	OK
WAS	OK

Resource Group

Name: Online Trading Application
Class: ResourceGroup
Automation domain: E2E_Domain
Node:
Owner:
Info link:
Open Service in TBSM
Description:

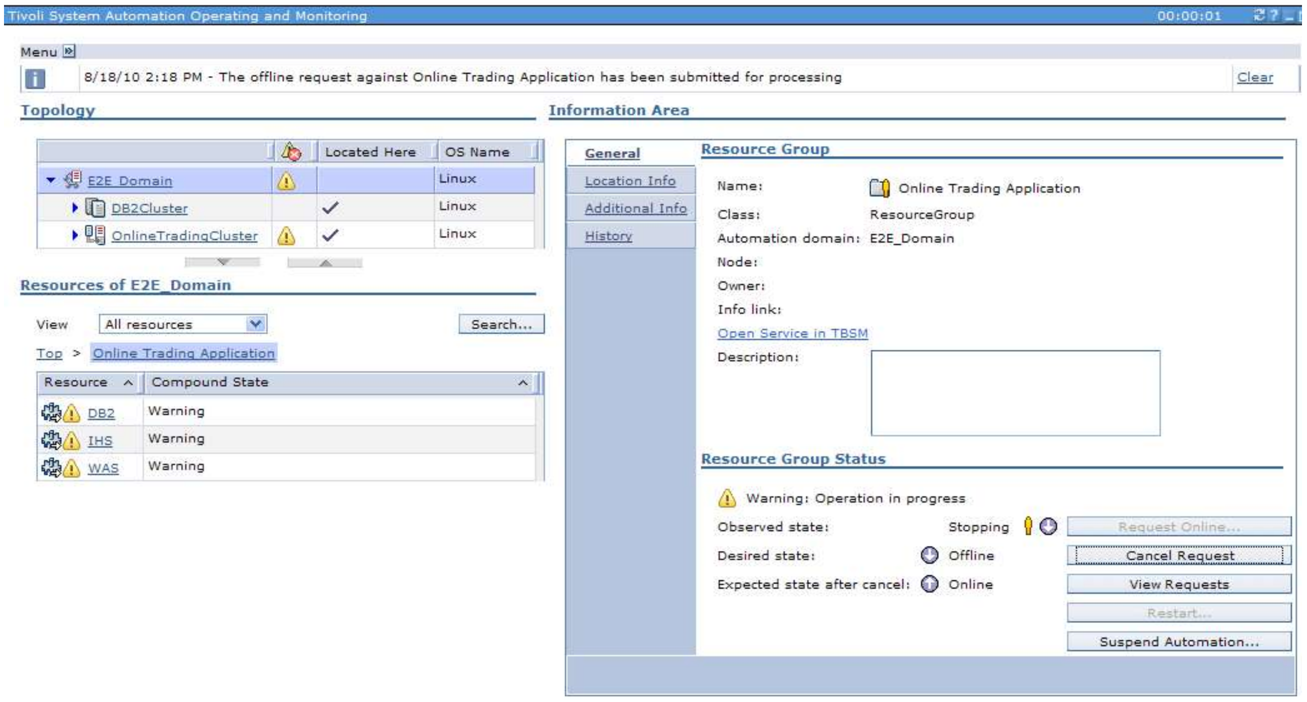
Resource Group Status

🟩 The resource works as desired
Observed state: Online
Desired state: Online

The SA Operations Console shows Topology information (Clusters/Systems) and information about the automated service instances. It shows the same service „Online Trading Application", which is already selected showing details about the service in the Information Area. The „Information Area" shows status information and gives operational control.

The operator now clicks on the „Request Offline" button to shutdown the whole “Online Trading Application“. The request will be processed by the Tivoli System Automation Application Manager automation engine, which stops all components in the correct sequence:

Tivoli System Automation Operations Console


Tivoli System Automation Operating and Monitoring

8/18/10 2:18 PM - The offline request against Online Trading Application has been submitted for processing

Topology

		Located Here	OS Name
E2E_Domain	⚠		Linux
DB2Cluster		✓	Linux
OnlineTradingCluster	⚠	✓	Linux

Resources of E2E_Domain

View: All resources

Top > Online Trading Application

Resource	Compound State
DB2	Warning
IHS	Warning
WAS	Warning

Information Area

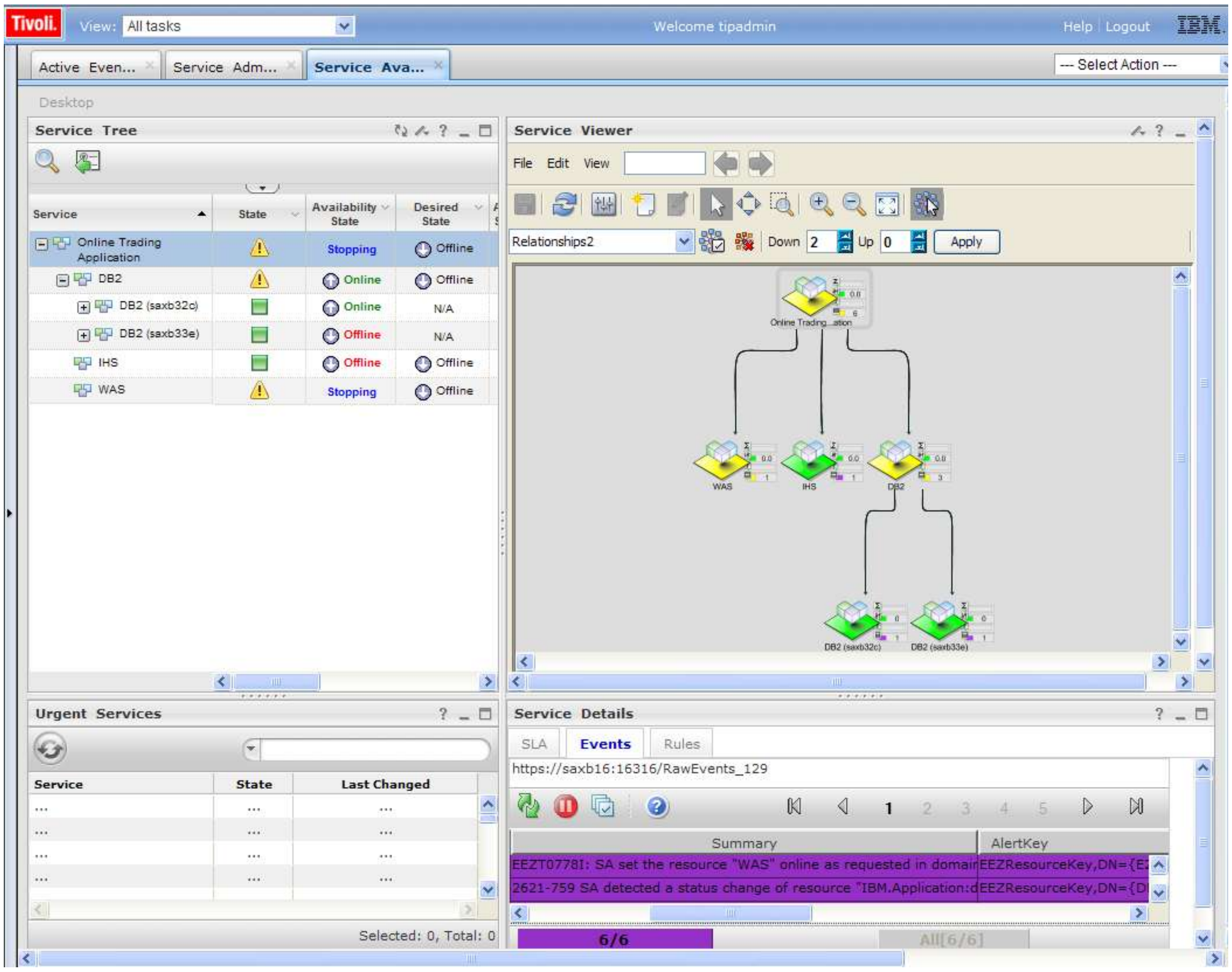
Resource Group

Name: Online Trading Application
Class: ResourceGroup
Automation domain: E2E_Domain
Node:
Owner:
Info link:
Open Service in TBSM
Description:

Resource Group Status

Warning: Operation in progress
Observed state: Stopping
Desired state: Offline
Expected state after cancel: Online

You can monitor this also in TBSM: Switch back to the „Service Availability" Workspace while the service components are being stopped. This can be done using the link “Open Service in TBSM", which opens the Service Availability Workspace and selects the Online Trading Application in the TBSM Service Viewer:

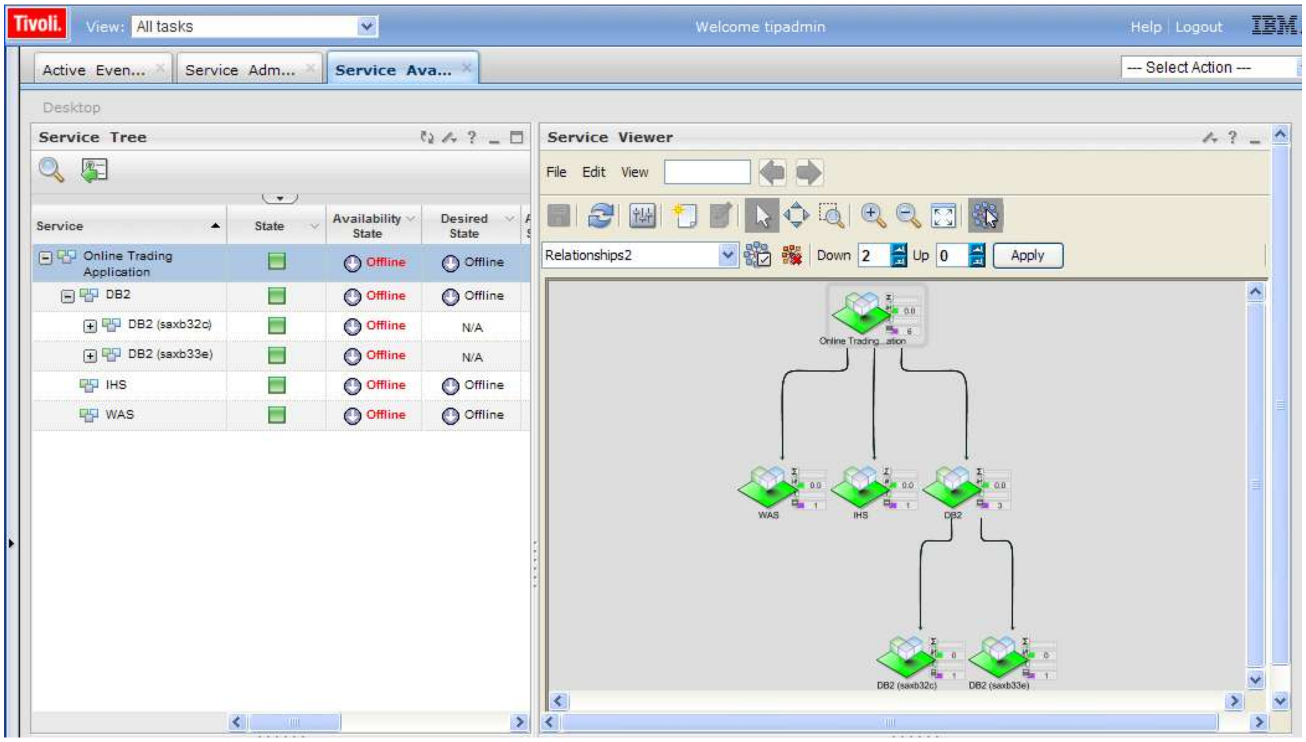


Service	State	Availability State	Desired State
Online Trading Application	⚠	Stopping	Offline
DB2	⚠	Online	Offline
DB2 (saxb32c)	🟩	Online	N/A
DB2 (saxb33e)	🟩	Offline	N/A
IHS	🟩	Offline	Offline
WAS	⚠	Stopping	Offline

https://saxb16:16316/RawEvents_129

Summary	AlertKey
EEZT0778I: SA set the resource "WAS" online as requested in domain	EEZResourceKey,DN={E
2621-759 SA detected a status change of resource "IBM.Application:d	EEZResourceKey,DN={D

In above screenshot you can see that IHS has already been stopped and now WebSphere Application Server is being stopped by Tivoli System Automation Application Manager. After WebSphere Application Server has fully stopped, Tivoli System Automation Application Manager then stops DB2. Finally, everything is shut down:



View: All tasks

Service	State	Availability State	Desired State
Online Trading Application	🟩	Offline	Offline
DB2	🟩	Offline	Offline
DB2 (saxb32c)	🟩	Offline	N/A
DB2 (saxb33e)	🟩	Offline	N/A
IHS	🟩	Offline	Offline
WAS	🟩	Offline	Offline

Relationships2 Down 2 Up 0

Again note that the state of the “Online Trading Application" and its components is “green" although the service is not running. The application shows as green because it was intentionally shut down by an operator, and it is not a failure situation. This knowledge can be delivered by using the SA Compound State to determine the overall state of service instances.

2 Part 2: Step-by-step setup instructions

2.1 Setting up event forwarding to IBM Tivoli Netcool/OMNIbus

This chapter describes how to set up IBM Tivoli Netcool/OMNIbus to forward Tivoli System Automation events to the OMNIbus event console, and how to set up launch-in-context support for OMNIbus. This OMNIbus setup is a prerequisite for the integration of System Automation Application Manager with Tivoli Business Service Manager.

2.1.1 Prerequisites

Because System Automation Application Manager and Tivoli System Automation for Multiplatforms use Tivoli Event Integration Facility (EIF) events for communication, the following components are required:

- IBM Tivoli Netcool/OMNIbus (OMNIbus)
- OMNIbus Probes Library for Nonnative Base
- OMNIbus Probe for Tivoli EIF (EIF Probe). This probe can receive EIF events sent from System Automation and forward them to the ObjectServer.

The following minimum product versions are required:

- OMNIbus Probe for Tivoli EIF V9.0
- IBM Tivoli Netcool/OMNIbus V7.2.1
- IBM Tivoli System Automation Application Manager V3.2.1
- IBM Tivoli System Automation for Multiplatforms V3.2.1

Note: If you are running IBM Tivoli Netcool/OMNIbus V7.2.1, you must have Interim Fix 3 (7.2.1.5-IF0003) installed. If you are running IBM Tivoli Netcool/OMNIbus V7.3 or later, you need no additional fix packs.

Install and set up these components according to the documentation available in the OMNIbus Information Center:
http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc_7.3.0/omnibus/wip/welcome.htm

Environment variables:

$NCHOME

Refers to the Netcool Home Directory into which the packages are installed. The default directory on Linux systems is /opt/IBM/tivoli/netcool.

$OMNIHOME

The $OMNIHOME variable is used to provide legacy support for scripts, third-party applications, and probes that continue to use the $OMNIHOME environment variable. $OMNIHOME refers to $NCHOME/omnibus.

2.1.2 Configuring OMNIbus to process System Automation events

2.1.2.1 Updating the OMNIbus database

The OMNIbus ObjectServer database includes the alerts.status table that contains all fields that are shown and selected by the Event-List-Viewer. For System Automation events, some additional columns have to be created in the alerts.status table.

The `sa_db_update.sql` file creates the new columns in the alerts.status table. The event class used for events from Tivoli System Automation is created as well. Tivoli System Automation uses the event class 87725 for its events. The class is used to associate tools like the launch-in-context tool, to a specific type of event.

Enter the following command on the OMNIbus server:

On UNIX® systems:

```
$OMNIHOME/bin/nco_sql -server NCOMS -username root < sa_db_update.sql
```

On Windows systems:

```
%NCHOME%\bin\redist\isql -S NCOMS -U root < sa_db_update.sql
```

Enter your password when prompted.

You can find the `sa_db_update.sql` file on the System Automation Application Manager product CD in the `/integration` directory.

Note: The event class 87725 is predefined in OMNIbus Version 7.3.1 or later. If you run the `sa_db_update.sql` script using OMNIbus Version 7.3.1, you receive the following error message:

```
ERROR=Attempt to insert duplicate row on line 2 of statement 'insert into
alerts.conversions values ( 'Class87725',’Class',87725,'Tivoli System
Automation' );...'
```

You can ignore this error message.

Verify that the SAspecific columns and event class have been successfully added to OMNIbus:

1. Open the Netcool/OMNIbus Administrator window by using the `nco_config` command.

2. From the Netcool/OMNIbus Administrator window, select the **System** menu button.
3. Click **Databases**. The Databases pane opens.
4. Select the **alerts.status** table. The alerts.status table pane opens.
5. Verify that the following columns are listed:
    a) SACompoundState
    b) SADesiredState
    c) SAObservedState
    d) SAOperationalState
    e) SADomainName
    f) SAResourceName
    g) SAReferencedResource
    h) SAEventReason
    i) SAExludedFromAutomation
    j) SADesiredRole
    k) SAObservedRole
    l) SADomainState
    m) SACommunicationState
6. From the Netcool/OMNIbus Administrator window, select the **Visual** menu button.
7. Click **Classes**. The Classes pane opens.
8. Verify that the class with ID **87725** and label **Tivoli System Automation** is listed in the table.

2.1.2.2 Enable rules file for System Automation

An OMNIbus rules file defines how the probe should process event data to create a meaningful alert. For each alert, the rules file also creates an identifier that uniquely identifies the problem source.

The Probe for Tivoli EIF uses a standard rules file named tivoli_eif.rules. System Automation Application Manager includes the System Automation-specific `tivoli_eif_sa.rules` rules file. This file must be be included within the tivoli_eif.rules default by using an include statement.

The `tivoli_eif_sa.rules` rules file processes an EIF event received by the Probe for Tivoli EIF if the 'source' event field contains the "SystemAutomation value".

The default tivoli_eif.rules file is located on the system where the Probe for Tivoli EIF is installed in the following directory:

On Windows® systems:

`%OMNIHOME%\probes\<os_dir>\tivoli_eif.rules`

On UNIX® systems:

`$OMNIHOME/probes/<os_dir>/tivoli_eif.rules`

Perform the following steps to enable the `tivoli_eif_sa.rules` file:

1. Copy the `tivoli_eif_sa.rules` file, which is located in the `/integration` directory on the System Automation Application Manager product CD to the system where the OMNIbus Probe for Tivoli EIF is installed. As a target directory, choose the directory where the tivoli_eif.rules file is located.

   1. Enable the included `tivoli_eif_sa.rules` rules file. To enable this file, edit the `tivoli_eif.rules` file that is used in the Probe for Tivoli EIF and add an **`include`** statement for the `tivoli_eif_sa.rules` file.

      The content of the `tivoli_eif.rules` file looks different depending on the type of OMNIbus installation you have:

      a) **If you are using a standalone OMNIbus installation:**

      Open the tivoli_eif.rules file in a text editor and add the include statement after the `switch ($source)` block:

```
:
else
{
        switch($source)
        {
                case "dummy case statement": ### This will prevent syntax errors in case
no includes are added below.

                include "tivoli_eif_tpc.rules"
                include "tivoli_eif_tsm.rules"

                # Uncomment the following line when using TADDM integration
                # This rules file is available in OMNIbus 7.3 and newer only
                # include "tivoli_eif_taddm.rules"

                default:
                        # Comment out the following line when not receiving events from TEC
                        include "tivoli_eif_default.rules"
        }
        include "tivoli_eif_sa.rules"
}
```

      b) **If you are integrating with Tivoli Business Service Manager (TBSM) and using the OMNIbus version included with TBSM:**

      Open the `tivoli_eif.rules` file in a text editor and add the include statement in the block where the other predefined rules files are included. Search for the line **`"# Include`**

`customer rules which would override any previous rules."` and add the include statement for `tivoli_eif_sa.rules` before this line:

```
:
:
###
### Handle TEC Events
###
include "tec_event.rules"

###
### Handle Z Events
###
# include "zos_event.rules"

###
### Handle Z user defined events.
###
# include "zos_event_user_defined.rules"

###
### Handle Z identity assignement.
###
# include "zos_identity.rules"

###
### Handle EE( Event Enablement) events.
###
# include "tivoli_eif_ee.rules"

include "tivoli_eif_sa.rules"

# Include customer rules which would override any previous rules.
# include "customer_override.rules"
:
:
```

3. Stop the EIF probe.

   On Windows systems:
   In the **Control Panel**, open **Administrative Tools**, and then **Services**. In the list of services, double-click the EIF probe, and then click **Stop**.

   On UNIX systems:
   Issue the following command from the command line:
   `$OMNIHOME/bin/nco_pa_stop -process <probe_name>`

4. Restart the EIF probe.

   On Windows systems:
   In the **Control Panel**, open **Administrative Tools**, and then **Services**. In the list of services, double-click the EIF probe, and then click **Start**.

   On UNIX systems :
   Issue the following command from the command line:
   `$OMNIHOME/bin/nco_pa_start —process <probe_name>`

2.1.3 Configuring launch-in-context support

The launch-in-context support for System Automation Application Manager allows navigating from a displayed event in the OMNIbus Active Event List to the corresponding resource or domain in the System Automation operations console.

Example usage scenario:

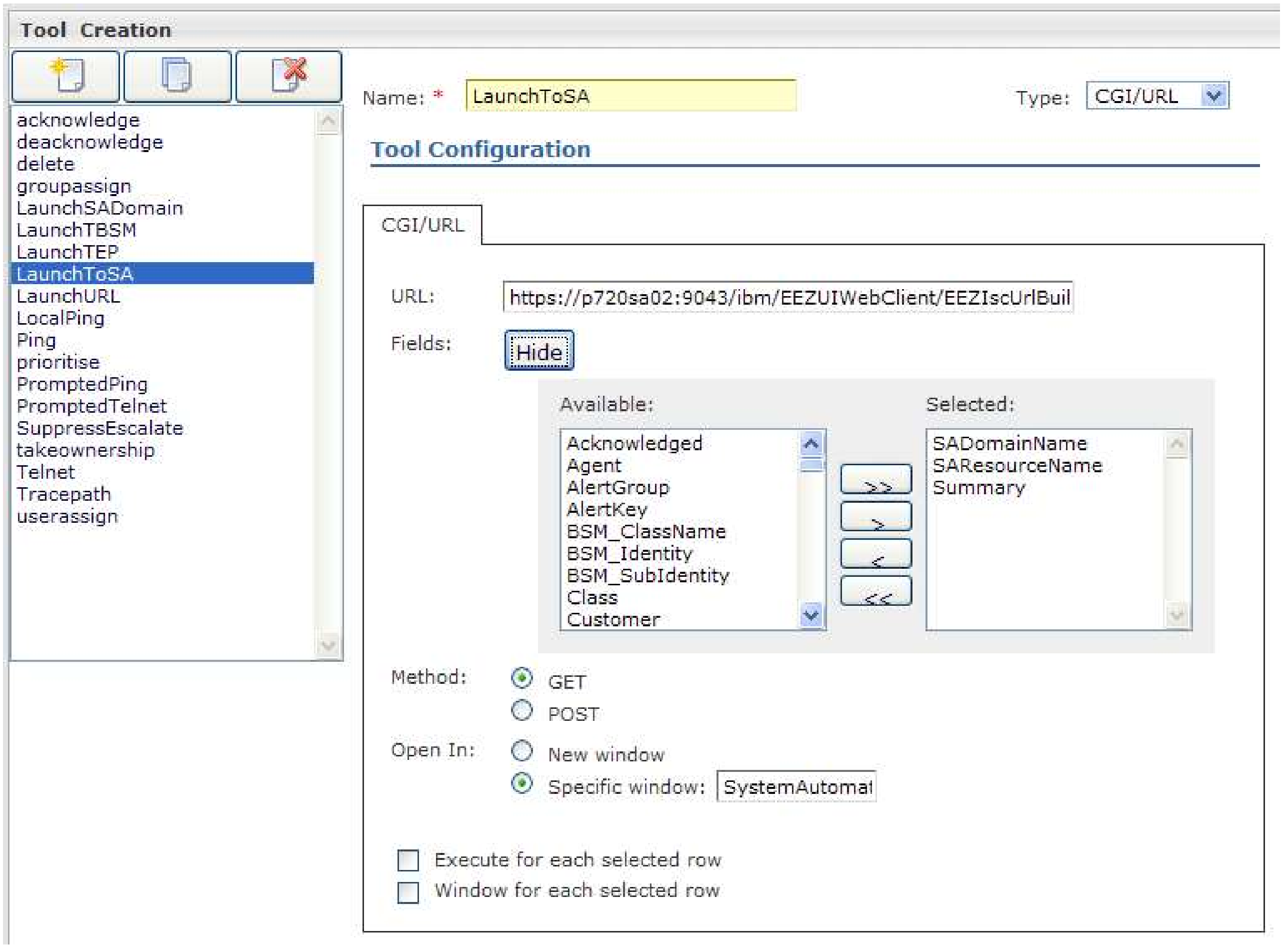
1. An application failure is detected by System Automation and a corresponding event is forwarded to OMNIbus.
2. An operator sees an error event in the OMNIbus Active Event List that indicates that an automated application has failed.
3. For further analysis, the operator selects the event and clicks on a menu entry to launch the SA Operations Console.
4. A new browser window is opened and connects to the ISC that automatically launches the SA Operations Console.
5. The SA Operations Console automatically navigates to the application that caused the error event and selects it.
6. The operator now sees all detailed status information and relationships that are provided by System Automation.

2.1.3.1 Creating the launch-in-context tool

Create a launch-in-context tool that launches the SA Operations Console from the event managed by the ObjectServer:

1. Log in to the Tivoli Integrated Portal (TIP) hosting the OMNIbus Web GUI as tipadmin.
2. Navigate to **Administration -> Event Management Tools -> Tool Creation**.
3. Select **Create Tool**.
4. Select **CGI/URL** as Type.
5. Enter **https://<ISC_HOST>:<ISC_PORT>/ibm/EEZUIWebClient/EEZIscUrlBuilder Servlet.** ISC_HOST is the server where the SA Operations Console is installed. ISC_PORT is the secure port for the Integrated Solutions Console hosting the Operations Console.
6. Click **Fields : Show** and select the following fields that should be passed as context arguments to the URL:
   a) SADomainName
   b) SAResourceName
   c) Summary
7. Select **Method: GET**

8. Select **Open In -> Specific Window** and type a Window name, for example, SystemAutomation. Launches from OMNIbus always use the same window to open the SA Operations Console.
9. In the Access Criteria section, select the **Class** tab and choose the "Tivoli System Automation" class from the list of available event classes. This step ensures that the launch-in-context tool is only available for events coming from System Automation.


Tool Creation

acknowledge
deacknowledge
delete
groupassign
LaunchSADomain
LaunchTBSM
LaunchTEP
LaunchToSA
LaunchURL
LocalPing
Ping
prioritise
PromptedPing
PromptedTelnet
SuppressEscalate
takeownership
Telnet
Tracepath
userassign

Name: * LaunchToSA
Type: CGI/URL

Tool Configuration

CGI/URL

URL: https://p720sa02:9043/ibm/EEZUIWebClient/EEZIscUrlBuil

Fields: Hide

Available: Acknowledged, Agent, AlertGroup, AlertKey, BSM_ClassName, BSM_Identity, BSM_SubIdentity, Class, Customer

Selected: SADomainName, SAResourceName, Summary

Method: [✓ GET] POST

Open In: New window [✓ Specific window: SystemAutomat]

☐ Execute for each selected row
☐ Window for each selected row

Sample configuration of launch entry

10. Click **Save** to save the new launch-in-context tool.

2.1.3.2 Defining the menu entry

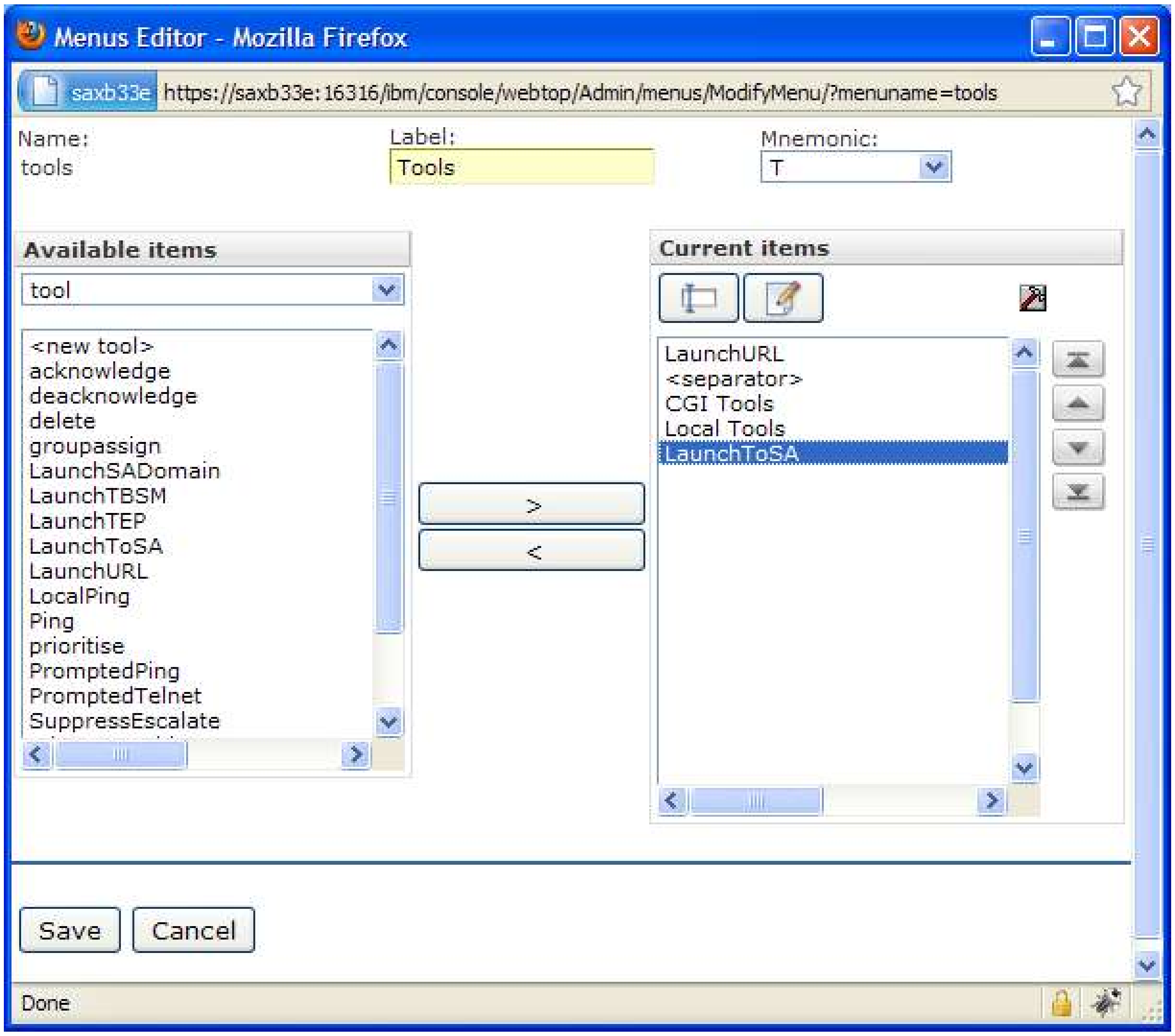
After the launch-in-context tool has been defined, you can configure the menu entry to display the launch-in-context tool.

To add the new launch-in-context tool to the Active Event List Tools menu, complete the following steps:

1. Navigate to **Administration -> Event Management Tools -> Menu Configuration -> tools ->**

Modify.

2. Select the new launch-in-context tool from the **Available items** list and add it to the list of **Current items**.
3. Click **Save** to store the modified configuration of the Tools menu.



Menus Editor - Mozilla Firefox
saxb33e https://saxb33e:16316/ibm/console/webtop/Admin/menus/ModifyMenu/?menuname=tools

Name: tools
Label: Tools
Mnemonic: T

Available items: tool

```
<new tool>
acknowledge
deacknowledge
delete
groupassign
LaunchSADomain
LaunchTBSM
LaunchTEP
LaunchToSA
LaunchURL
LocalPing
Ping
prioritise
PromptedPing
PromptedTelnet
SuppressEscalate
```

Current items:

```
LaunchURL
<separator>
CGI Tools
Local Tools
LaunchToSA
```

Adding launch-in-context tool to Tools menu

2.1.4 Enabling event generation in System Automation Application Manager

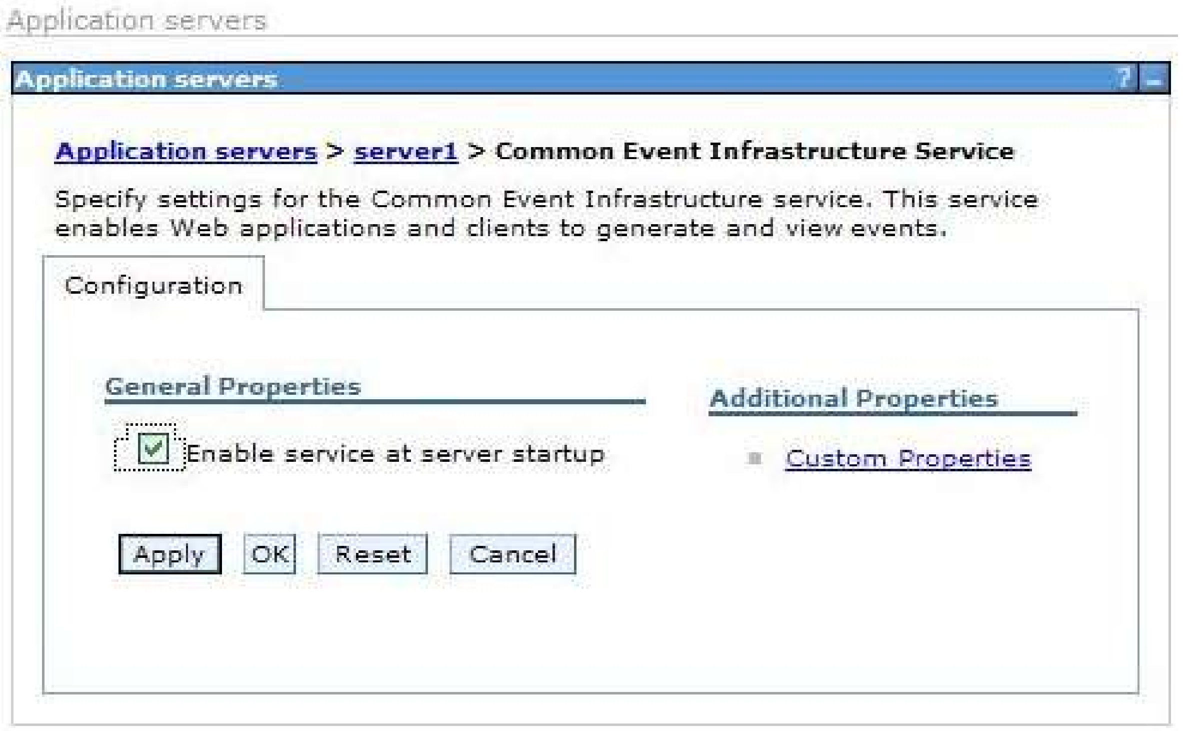
If you want to send System Automation events to OMNIbus, you must enable event forwarding in System Automation Application Manager.

If you have not activated or configured the EIF Event generation and forwarding function during the installation of System Automation Application Manager, you can do so by completing the following steps in the Integrated Solutions Console:

1. Activate the Common Event Infrastructure (CEI) service when the application server is started:

   a) Click **Servers** > **Application servers** > *<server_name>* > **Container Services** > **Common Event Infrastructure Service**.

   b) Click the **Enable service at server startup** check box.



Application servers

Application servers

Application servers > server1 > Common Event Infrastructure Service

Specify settings for the Common Event Infrastructure service. This service enables Web applications and clients to generate and view events.

[✓] Enable service at server startup

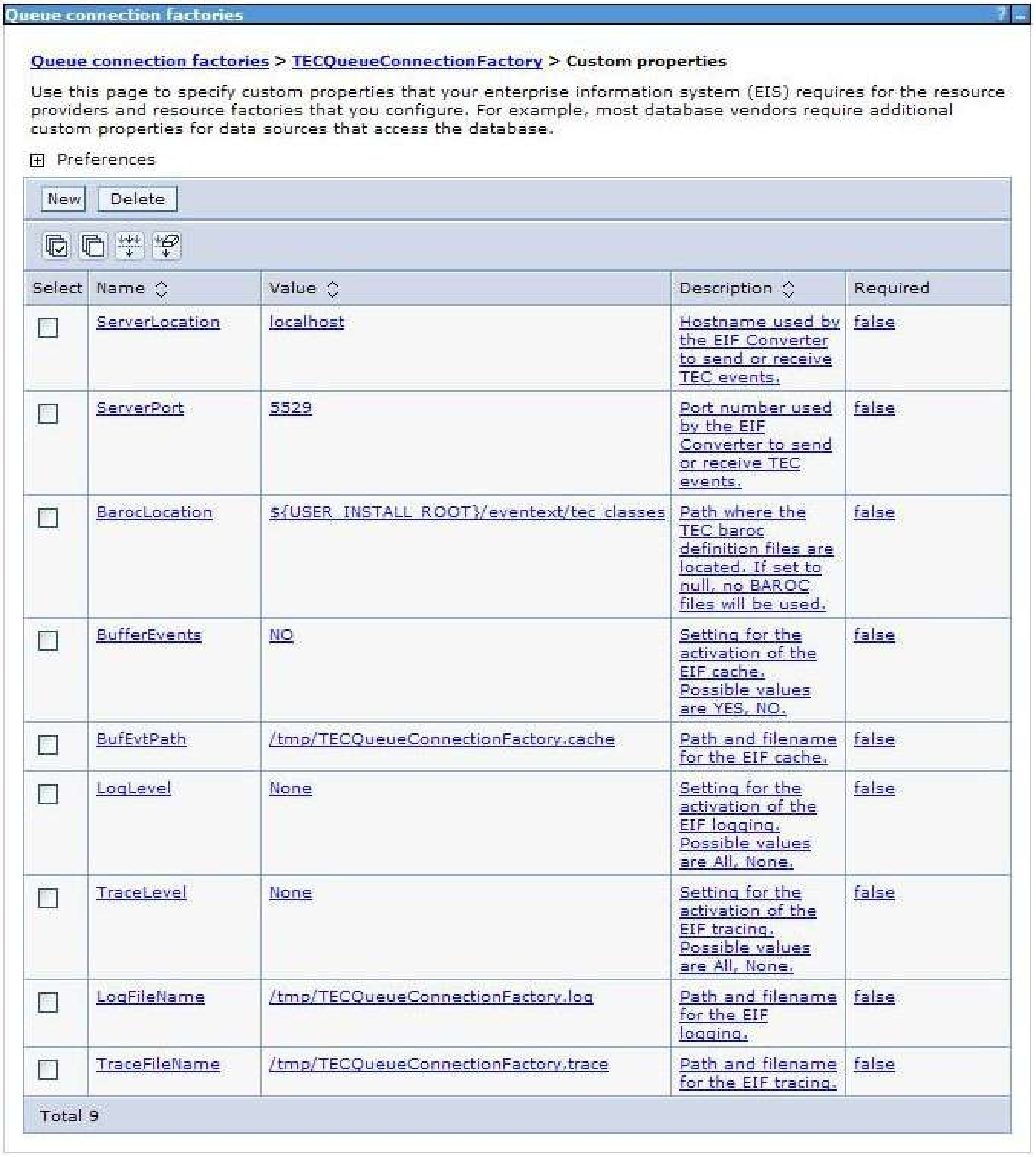
   c) Save the Master configuration and restart the WebSphere Application Server.

2. Configure the Event Server:

   a) In the Integrated Solutions Console, click **Resources > JMS > Queue connection factories** > **TECQueueConnectionFactory** > **Custom properties**.

   b) Set **ServerLocation** to the system where the OMNIbus Probe for Tivoli EIF is installed.

   c) Set **ServerPort** to the port the OMNIbus Probe for Tivoli EIF is listening to. Typically, this is port 5529.



Queue connection factories

Queue connection factories > TECQueueConnectionFactory > **Custom properties**

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

⊞ Preferences

Select	Name	Value	Description	Required
☐	ServerLocation	localhost	Hostname used by the EIF Converter to send or receive TEC events.	false
☐	ServerPort	5529	Port number used by the EIF Converter to send or receive TEC events.	false
☐	BarocLocation	${USER_INSTALL_ROOT}/eventext/tec_classes	Path where the TEC baroc definition files are located. If set to null, no BAROC files will be used.	false
☐	BufferEvents	NO	Setting for the activation of the EIF cache. Possible values are YES, NO.	false
☐	BufEvtPath	/tmp/TECQueueConnectionFactory.cache	Path and filename for the EIF cache.	false
☐	LogLevel	None	Setting for the activation of the EIF logging. Possible values are All, None.	false
☐	TraceLevel	None	Setting for the activation of the EIF tracing. Possible values are All, None.	false
☐	LogFileName	/tmp/TECQueueConnectionFactory.log	Path and filename for the EIF logging.	false
☐	TraceFileName	/tmp/TECQueueConnectionFactory.trace	Path and filename for the EIF tracing.	false

Total 9

3. Start the end-to-end automation configuration tool (cfgeezdmn). On the Domain page, select the check box "Enable EIF event generation for TEC or OMNIbus".

4. Restart the WebSphere Application Server and the end-to-end automation engine (eezdmn).

2.1.5 Enabling event generation in System Automation for Multiplatforms

If you want to send System Automation events for resources managed by System Automation for Multiplatforms to OMNIbus, you must enable event forwarding in Tivoli System Automation for Multiplatforms too. This step is required if you want to integrate resources into TBSM that are located within a High Availability Cluster managed by Tivoli System Automation for Multiplatforms, for example, a DB2 instance located on a specific node.

You activate and configure the EIF Event generation and forwarding function by enabling the TEC publisher. Do this as follows:

1. Copythe `/usr/sbin/rsct/samples/tec/TECPublisher.conf` files into `/etc/Tivoli/tec` on all System Automation for Multiplatforms cluster nodes.
2. Customize the `/etc/Tivoli/tec/samPublisher.conf` publisher configuration file and the `/etc/Tivoli/tec/TECPublisher.conf` EIF configuration file on each cluster node.
3. Enable the publisher with the `samctrl -e TEC` command on a System Automation for Multiplatforms cluster node. By default, the publisher is disabled. Use the **`samctrl -e P`** command to enable all Publishers specified in the Publisher configuration file at once.

2.1.5.1 Publisher configuration file

The `/etc/Tivoli/tec/samPublisher.conf` publisher configuration file specifies a list of all target consumers and their parameters. This file uses the following syntax format:

```
# Publisher configuration file
# file name: /etc/Tivoli/tec/samPublisher.conf
#
# File format:
#    <keyword>=<value>
#
#   Publisher     - unique name of the publisher
#                   name length: 1-8 characters
#                   valid characters: '0'-'9', 'A'-'Z', 'a'-'z' and '_'
#   LibraryPath   - name of the publisher library
#   ConfigPath    - full path to the configuration file
#
# Multiple entries of the Publisher, LibraryPath and ConfigPath can be specified.
# One triplet for each publisher target consumer.
# Maximum supported publishers: 15

# Online Update section -----------------------------------------------------
# End Online Update section -------------------------------------------------

Publisher=TEC
LibraryPath=libTECPublisher.so
ConfigPath=/etc/Tivoli/tec/TECPublisher.conf

# Publisher=TEC2
# LibraryPath=libTECPublisher.so
# ConfigPath=/etc/Tivoli/tec/TECPublisher2.conf
```

To enable the Tivoli Enterprise Console publisher function, make sure that the entry for Publisher=TEC is defined in the `/etc/Tivoli/tec/samPublisher.conf` file. Entries for other publishers may also be defined in this file. Do not modify or remove any of those entries. Also keep in mind that the Replicate function of the configuration utility for the end-to-end automation adapter of System Automation for Multiplatforms replicates the `/etc/Tivoli/tec/samPublisher.conf` file to the other nodes in the automation domain.

2.1.5.2 About the EIF configuration file /etc/Tivoli/tec/TECPublisher.conf

The EIF configuration file specifies all parameters required to connect to a specific OMNIbus server. The file name must match the name specified as the "ConfigPath" parameter in the publisher configuration file.

The syntax format of the EIF configuration file for the Tivoli Enterprise Console publisher is the existing Tivoli Enterprise Console EIF configuration file syntax.

```
# IBM_PROLOG_BEGIN_TAG
# This is an automatically generated prolog.
#
#
#
# Licensed Materials - Property of IBM
#
# Restricted Materials of IBM
#
# (C) COPYRIGHT International Business Machines Corp. 2003,2010
# All Rights Reserved
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# IBM_PROLOG_END_TAG
#
# TEC EIF configuration file
#
# File format:
#    <keyword>=<value>
#
#   ServerLocation      - name of the host where the TEC server or the OMNIbus
#                         Probe for Tivoli EIF is running
#   ServerPort          - port number on which the TEC server or the OMNIbus Probe
#                         for Tivoli EIF is listening to EIF events.
#                         TME TEC events are not supported.
#                         5529 - default non-TME port for TEC servers on Windows
#                                and for the OMNIbus Probe for Tivoli EIF
#                         0    - default non-TME port for TEC servers on AIX and Linux
#   ConnectionMode      - connection_oriented OR connection_less
#                       - (default is connection_oriented)
#   BufferEvents        - Specifies is the event buffering cache file is enabled
#                         (YES | MEMORY_ONLY | NO) (default is YES)
#   BufEvtPath          - Specifies the full path name of the cache file
#                         (default: /etc/Tivoli/tec/cache)
#   NO_UTF8_CONVERSION  - Specifies if UTF8 conversion is done again in EIF library
#                         Must be YES, otherwise event is corrupted (default is NO)
#   FilterMode          - Specifies whether events that match a Filter are
#                         sent to the event server (FilterMode=IN)
#                         or are discarded (FilterMode=OUT) (default is OUT)
#   Filter              - Filter:Class=class_name[;attribute=value]*
#
# For the description of all supported keywords and there values see
# Book: "Tivoli Event Integration Facility User's Guide", GC32-0691
# Chapter: "Appendix B. Keywords for Configuration Files"
# Put the server name or IP address of the server on which the TEC or the OMNIbus
# Probe for Tivoli EIF is running into the "ServerLocation" field.
ServerLocation=omnibushost.ibm.com
ServerPort=5529
ConnectionMode=connection_less
BufferEvents=YES
BufEvtPath=/etc/Tivoli/tec/TECPublisher.cache
NO_UTF8_CONVERSION=YES

# Default Filters
#
# The filters below will filter all resource status events with serverity HARMLESS
# and serverity WARNING. In addition all configuration events will be filtered.
# Disable a filter with a pound key at the beginning of the line.
# To activate a configuration file change use the command "samctrl" to disable and enable a
publisher.
#
# Filter all relationship add / delete events
Filter:Class=SystemAutomation_Relationship_Configuration_Change
# Filter all resource add / delete events
Filter:Class=SystemAutomation_Resource_Configuration_Change
# Filter resource status events with serverity HARMLESS
# Filter:Class=SystemAutomation_Resource_Status_Change;severity=HARMLESS
# Filter resource status events with serverity WARNING
# Filter:Class=SystemAutomation_Resource_Status_Change;severity=WARNING
# Filter all request add / delete events
# Filter:Class=SystemAutomation_Request_Configuration_Change
```

For ServerLocation and ServerPort, specify the server name and port information of the OMNIbus Probe for Tivoli EIF.

To avoid flooding the OMNIbus server with a huge amount of messages, filters are provided in the EIF configuration file. By default, all filters are enabled, which results in only critical messages being sent to OMNIbus.

However, the TBSM integration relies on status change events being sent to OMNIbus. Therefore, you have to disable the filter for SystemAutomation_Resource_Status_Change and SystemAutomation_Request_Configuration_Change events by using the comment character #.

2.1.5.3 Enabling the publisher

The Publisher function is disabled by default. To query the status of the publisher, issue the following command:

```
node1:/usr/sbin/rsct/samples/tec # lssamctrl
```

The following System Automation for Multiplatforms control information is displayed:

```
SAMControl:

     TimeOut                    = 60
     RetryCount                 = 3
     Automation                 = Auto
     ExcludedNodes              = {}
     ResourceRestartTimeOut     = 5
     ActiveVersion              = [3.2.0.0,Wed Feb 17 20:19:07  2010]
     EnablePublisher            = TEC EEZ
     TraceLevel                 = 31
     ActivePolicy               = []
     CleanupLsist               = {}
     PublisherList              = {}
```

The attribute EnablePublisher lists all enabled publishers.

To enable the Tivoli Enterprise Console publisher, issue this command on any node:

```
node1:/usr/sbin/rsct/samples/tec # samctrl -e TEC
```

To disable the Tivoli Enterprise Console publisher, issue this command on any node:

```
node1:/usr/sbin/rsct/samples/tec # samctrl -d TEC
```

2.2 Setting up Tivoli Business Service Manager (TBSM)

2.2.1 Prerequisites

To integrate System Automation resources in TBSM, the following prerequisites must be met:

- Configure OMNIbus to process Tivoli System Automation Application events (see "Configuring OMNIbus to process System Automation events" for details).
- Set up event forwarding to OMNIbus for System Automation Application Manager events (see "Enabling event generation in System Automation Application Manager" for details).
- If you want to include System Automation for Multiplatforms resources in TBSM service trees, set up event forwarding to OMNIbus for System Automation for Multiplatforms events (see "Enabling event generation in System Automation for Multiplatforms" for details).
- Install Tivoli Business Service Manager (TBSM) V.4.2.1 or later.
- Update the Netcool OMNIbus ObjectServer schema for TBSM.
  - If you have an existing OMNIbus server, import the `tbsm_db_update.sql` and `ClearServiceDeps.auto` schema files.
  - If OMNIbus is installed with TBSM, the TBSM installer completes the required schema updates.

You can find all TBSM-specific product information in the IBM Tivoli Business Service Manager Information Center. For more information about the installation of the product, go to http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/welcome.htm.

2.2.2 Configuring TBSM

To simplify the process for defining and configuring services in TBSM, ***service templates*** can be defined for services instances with common behavior. Rather than define each of the services and their dependencies individually, one template can be created for a type of service and then be assigned to applicable services.

Service instances represent actual services that are assigned a template. The template defines how a service responds to incoming data and the status of other services. Services of the same type should be assigned to a common template. This assignment allows using the same template rules to evaluate the status of multiple services.

When you assign a template to a service, you tag the service with the template. Templates eliminate the necessity of creating the same rules for a service type more than once.

2.2.2.1 About the System Automation Service Template for TBSM

The Tivoli System Automation Application Manager provides a TBSM Service Template that is used for System Automation resources that are displayed in a TBSM service tree. The Service Template is named EEZ_SystemAutomationResource. It provides the following functions:

- An Incoming Status Rule named SACompoundState that uses state change events coming from System Automation resources to determine the overall state of services
- Text-based Incoming Status Rules that export the System Automation Observed State and other System Automation specific states of a resource so that they can be used in TBSM Views. See the following "Customizing TBSM views to add information from System Automation" section for more information how to use the text-based incoming status rules.
- Additional properties that allow launch-in-context to the System Automation Operations Console. See the following "Configuring Launch-in-Context Support" section for more information.

Incoming Status Rule SACompoundState

The EEZ_SystemAutomationResource Service Template contains an incoming status rule named SACompoundState that is used to determine the overall state of a service. If the Service Template has been assigned to a specific service instance, resource state change events coming from System Automation will influence the overall state of the service. Events are associated with a service instance if the AlertKey in the event matches the AlertKey defined as Identifier for the Service Instance.

TBSM has three available overall states (Bad, Marginal, Good). The following mapping is defined in the SACompoundState rule to map resource state change events from System Automation to an overall TBSM state for a service instance:

Event Severity	TBSM State
5 (Critical)	Bad (Red)
4 (Major)	Bad (Red)
3 (Minor)	Marginal (Yellow)
1 (Indeterminate)	Good (Green)

Because of a one-to-one mapping from a resource's compound state to the event severity, the System Automation Compound State directly determines the TBSM state. The following table shows how the SA Compound State value determines the Event Severity of resource status change events:

SA Compound State	OMNIbus „Severity“ field
Fatal	5 (Critical)
Error	4 (Major)
Warning	3 (Minor)
OK	1 (Indeterminate)

2.2.2.2 Defining the System Automation service template in TBSM

The EEZ_SystemAutomationResource template is required to use System Automation events in TBSM. Import the EEZ_SystemAutomationResource template into TBSM as follows:

1. Copy the `EEZ_SystemAutomationResource.radsh` file from the `/integration` directory of the System Automation Application Manager product CD to a temporary directory where the TBSM data server is installed.
2. Open a command prompt on the TBSM data server system. Change to the directory where you have copied the `EEZ_SystemAutomationResource.radsh` file and issue the following command:

   On UNIX systems:
   ```
   cat EEZ_SystemAutomationResource.radsh | $TBSM_HOME/bin/rad_radshell
   ```

   On Windows systems:
   ```
   type EEZ_SystemAutomationResource.radsh | %TBSM_HOME%\bin\rad_radshell
   ```

The service template provided by System Automation is now defined in TBSM.

2.2.2.3 Defining trigger in Netcool/OMNIbus

In the OMNIbus ObjectServer, a new state change event for a resource replaces the previous event for that resource. This state change event is called *event deduplication*.

By default, TBSM processes only a deduplicated event when the value of the Severity field changed. In these cases, TBSM processes the deduplicated events and updates the service status accordingly. A status change is possible for a Tivoli System Automation resource that updates status fields used in the text-based incoming status rules that are contained in the EEZ_SystemAutomationResource service template. But the severity value does not change because the compound state of the resource does not change. Define a trigger in OMNIbus to ensure that TBSM updates the services in these cases as well.

The `sa_db_tbsm_update.sql` file is used to define the `update_tbsm_service_on_sa_events` trigger in OMNIbus. This trigger ensures that TBSM reprocesses events if one of the states that are used in the text-based incoming status rules changes, even if the severity value does not change. Whenever you want to use the text-based incoming status rules included in the EEZ_SystemAutomationResource service template, create this trigger definition.

Enter the following command on the OMNIbus server to define the trigger:

On Windows systems:
```
%NCHOME%\bin\redist\isql -S NCOMS -U root < sa_db_tbsm_update.sql
```

On UNIX systems :
```
$OMNIHOME/bin/nco_sql -server NCOMS -username root < sa_db_tbsm_update.sql
```

Enter the password when prompted.

The `sa_db_tbsm_update.sql` is included with System Automation Application Manager and can be found in the `/integration` directory on the product CD.

2.2.3 Configuring the Discovery Library Toolkit

System Automation Application Manager provides a Discovery Library Adapter (DLA) to export the currently active System Automation Application Manager resource topology to an Identity Markup Language (IdML) discovery book.

You can use the Discovery Library Toolkit of TBSM to create TBSM service models from such an IdML book.

Using the Discovery Library Adapter automates the following tasks:

- Imports the System Automation Application Manager resources into the service model of TBSM. The new service instances show the grouping hierarchy defined in the end-to-end automation policy.
- Assigns the System Automation Service Template containing the incoming status rules to those services.
- Matches System Automation Application Manager events with the TBSM service instance without manual configuration.
- Defines the launch-in-context parameters of a service instance to enable launching to the SA

Operations Console in context of the service instance.

IdML books are XML files in the Identity Markup Language (IdML) format. These XML files conform to the IBM Common Data Model (CDM) and can be generated by a variety of systems and applications. They are used by the Discovery Library Toolkit to discover resources and relationships in a given environment.

This chapter describes how you can set up the Discovery Library Toolkit to enable TBSM to discover a service tree based on the active System Automation Application Manager resource topology.

Make sure the following products are installed and configured, before an IdML Book generated by System Automation Application Manager can be imported into TBSM:

- Install the Discovery Library Toolkit of TBSM.
- Configure TBSM to work with the Discovery Library Toolkit. Ensure that the `BSM_Templates.radsh` and `EEZ_SystemAutomationResource.radsh` files have been run and that the templates and policies have been added to your TBSM configuration. For more information, go to http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/welcome.

To correctly manage the data from System Automation Application Manager, modify the following XML control files on the TBSM server:

- Define a template mapping for the EEZ_SystemAutomationResource template in the `CDM_TO_TBSM4x_MAP_Templates.xml` file.
- Define an Event Identification Rule for System Automation events in the `EventIdentifierRules.xml` file.
- Make sure that importing of relationships except of group relationships is disabled by specifying a rule in `CDM_TO_TBSM4x_MAP.xml` file

You can find the three xml customization files in the following directory:

`$TBSM_HOME/XMLToolkit/xml`

2.2.3.1 Defining template mapping

To automatically assign the EEZ_SystemautomationResource template to the imported service instances while the IdML-Book is imported, complete the following configuration steps:

1. Open the `CDM_TO_TBSM4x_MAP_Templates.xml` file in a text editor.
2. Search for the <**template primary='BSM_BusinessService'**> line.
3. Add <**othertemplate name='EEZ_SystemAutomationResource'**/> to the XML

element. The added line is marked in bold:

```
<template primary='BSM_BusinessService' >
    <othertemplate name='SCR_RetrieveDependentObjectsTemplate'/>
    <othertemplate name='SCR_ServiceComponentRawStatusTemplate'/>
    <othertemplate name='EEZ_SystemAutomationResource'/>
    <cdmclass name='cdm:sys.BusinessSystem'/>
    <cdmclass name='cdm:process.BusinessService'/>
</template>
```

This change automatically assigns the EEZ_SystemAutomationResource service template to all services of the resources in the cdm:sys.BusinessSystem class that are imported using the DLA toolkit.

2.2.3.2 Customize TBSM dependency generation

The System Automation Application Manager DLA stores group and other relationships such as start or stop dependencies into the IdML book. Group relationships are stored by using the "cdm:federates" relation type , whereas non-group relationships such as start or stop dependencies are stored by using the "cdm:uses" relation type. If you want to import only the group hierarchy but not the non-group relationships into the service model of TBSM, you must disable the generation of dependencies for the "cdm:uses" relation type. This step simplifies the graphical representation of the resulting service graph in TBSM.

Complete the following steps to disable the generation of dependencies between service instances for non-group relations:

1. Open the `CDM_TO_TBSM4x_MAP.xml` file in an text editor.
2. Add the following line within the <Relationshiplnfo> tag:

```
<class type='cdm:uses'  isDependency='false' priority='medium'
source='cdm:sys.BusinessSystem' target='cdm:sys.BusinessSystem' />
```

After you added this line, the DLA toolkit no longer generates dependencies between imported service instances for the cdm:uses non-group relationship type.

2.2.3.3 Customizing event identification

To associate any event with a service instance imported from an IdML book, identification fields are required. If an event is received that has a matching field name and value as the service instance, the event is applied to the instance.

Status rules provided by System Automation use the **AlertKey** field for service identification. All System Automation events have an AlertKey field set with a unique source token value that represents a resource key. The resources contained in an IdML book created by System Automation Application Manager contain a source token attribute as well.

Define a policy to match System Automation events with the corresponding service instance by setting the AlertKey field to the source token value of the resource.

The generation of TBSM services identification field value pairs is implemented by defining rules in the `EventIdentifierRules.xml` file.

1. Open the `EventIdentifierRules.xml` file in a text editor.
2. Add the following Policy and Mapping elements to this file:

```
<Policy name="SystemAutomationApplicationManager">
 <Rule name="SourceTokenAsAlertKey" field="AlertKey">
 <Token keyword="SOURCETOKEN" value='IBM Tivoli System Automation Application Manager' />
 </Rule>
</Policy>
<Mapping policy="SystemAutomationApplicationManager" class="cdm:sys.BusinessSystem" />
```

When importing a resource of class cdm:sys.BusinessSystem, this policy sets the AlertKey field for all status rules of the corresponding service to the System Automation Application Manager source token value.

As a result, each TBSM service instance created from a System Automation Application Manager IdML Book has a unique AlertKey field value. Whenever TBSM detects an event with this AlertKey field value, it checks the event for status information for the associated service.

After you entered the changes to the XML control files, restart the TBSM Discovery Library Toolkit service to activate the changes.

2.2.4 Integrating System Automation resources in TBSM

This chapter describes how you can add System Automation resources to a TBSM service tree.

For resources that are contained in the end-to-end automation policy, you can make use of the DLA to automatically import these resources into TBSM. This procedure is described in the "Importing resources automatically" section.

If you want to add System Automation resources to TBSM that are not part of the end-to- end automation policy, for example, resources managed by System Automation for Multiplatforms, you must manually create a service instance in TBSM and then assign the System Automation Service Template. This procedure is described in the "Manually assigning the service template to a service instance" section. You also do this if you want to enrich service instances that already exist in a TBSM service tree with information from System Automation events.

2.2.4.1 Importing resources automatically

2.2.4.1.1 Generating a System Automation Application Manager IdML book

Use the **eezdla** command to create IdML books for System Automation Application Manager. Complete the following steps:

1. Log on to the system where the automation manager is running. On Windows Server 2008 systems, log on with the user credentials used to install the automation manager.
2. Enter **eezdla**.

The created IdML book is stored at the location that you configured on the Discovery Library Adapter tab in the end-to-end configuration tool (cfgeezdmn). On the same tab, you can configure the DLA to copy the IdML book to a remote TBSM server.

Naming conventions:

Names of IdML books that are created by System Automation Application Manager start with the EEZ application code, followed by the host name of the system where the automation manager is installed, and an UTC timestamp.

Example:

```
EEZ3210.e2esrvl.friendly.com.2010-02-15T14.57.47.093Z.refresh.xml
```

2.2.4.1.2 Importing a System Automation Application Manager IdML book into TBSM

lf the Discovery Library toolkit service is running, it automatically consumes IdML books available in a specific directory on the TBSM server. You can configure the specific directory by using the Discovery Library toolkit. The default directory for the location of IdML books is:

On Windows systems:
```
%TBSM_HOME%\discovery\dlbooks
```

On UNIX systems:
```
$TBSM_HOME/discovery/dlbooks
```

The System Automation Application Manager DLA can either be configured to automatically save its IdML books in that directory on the TBSM server, or the resulting IdML book must be manually copied to that location.

The discovery library toolkit service consumes the IdML book from System Automation Application Manager and populates the service component registry with System Automation Application Manager resources. The processing of the IdML book can be validated by checking the `msgGTM_XT.log` file for a message that is similar to the following line:

```
GTMCL5290I: Book EEZ321O.p720sa02.boeblingen.de.ibm.com.
2010-07-22T15.01.47.711Z.refresh.xmlprocessed successfully .
```

The following path indicates the default location of the log file:

On Windows systems:
```
%TBSM_HOME%\XMLtoolkit\log
```

On UNIX systems:
```
$TBSM_HOME/XMLtoolkit/log
```

Note: The DLA Toolkit service can be started in the following way:

- **On Windows systems:**
  Start the Discovery Library Toolkit service from the Services window or by issuing the net start ASICRToolkitSvc command.
- **On UNIX systems:**
  Start the Discovery Library Toolkit daemon by running the `tbsmrdr_start.sh` script in the `$TBSM_HOME/XMLtoolkit/bin` directory. The daemon is added to the `etc/init.d` directory and is automatically restarted if you restart the TBSM host.
-

2.2.4.1.3 Verifying imported services

You can use the following procedure to verify that the service instances are imported into the Service Component Registry correctly:

1. In the Tivoli Integrated Portal console task list, select **Administration** > **Service Administration**.
2. In the **Service Navigation** portlet, select **Service Component Repository** from the pull-down menu.
3. Expand **Component Repository** > **Business Services**. The imported end-to-end automation domains and resources are displayed below the Business Services entry. You can expand the domains and resource groups to verify that the group hierarchy has been correctly created.
4. Expand the end-to-end automation domain corresponding to the IdML book that has been imported. Select one of the contained top-level resources.
5. Select the **Edit Service** tab in the **Service Editor** portlet to edit the service.
6. Select the **Templates** tab and verify that the service template EEZ_SystemAutomationResource is listed in the "Selected Templates" list.
7. Select the **Identification Fields** tab and verify that for the incoming status rules of the EEZ_SystemAutomationResource template, the **AlertKey** field has a value corresponding to the resource's source token. For example,
   EEZResourceKey,DN= {E2E_Domain},NN= {},RN={Online Trading Application},RC= {ResourceGroup}
8. Select the **Additional** tab and verify that the sourceContactlnfo and sourceToken attributes for System Automation Application Manager are set correctly.
9. If the services have been imported correctly, a resource status event coming from System Automation Application Manager for those services influences the TBSM state displayed in the service tree. You can test this, for example, by stopping a group member defined in the end-to-end automation domain using the SA Operations Console. The group should then show a Warning state in TBSM based on the compound state changes of the group.

2.2.4.1.4 Adding imported services to a service model

When the System Automation Application Manager IdML 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 the service model:

1. In the Tivoli Integrated Portal console task list, select **Administration** > **Service Administration**.
2. In the **Service Navigation** portlet, create a new top-level service instance that you will use as a container for the imported System Automation services, or select an existing service instance as a parent of the new System Automation resources.
3. Select the **Edit Service** tab in the **Service Editor** portlet to edit this service.
4. Select the **Dependents** tab and click the **Add from Service Component Repository** button. The Service Tree window opens.
5. Expand **Component Registry** > **Business Services** > *your end-to-end automation domain*
6. Select the top-level resource groups that you want to add to your service model. These groups are added as children of the container service that you created in step 2.
7. When you have finished adding services, close the Service Tree pop-up window and save the service instance.

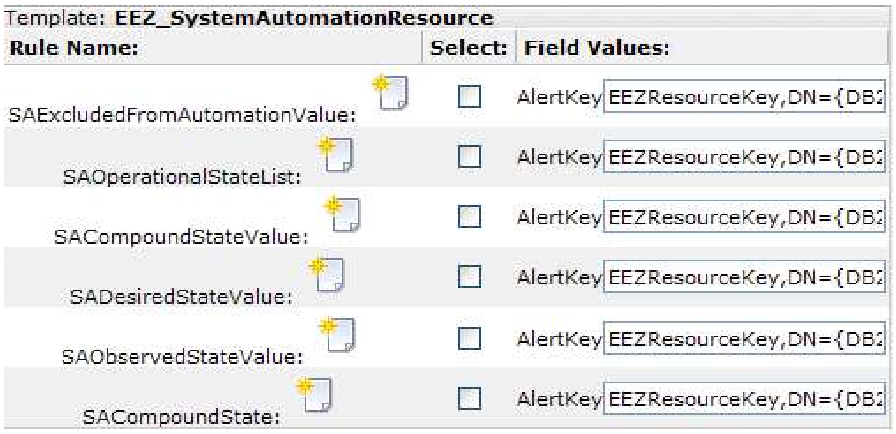
2.2.4.2 Manually assigning the service template to a service instance

If you want to add System Automation resources to TBSM that are not part of the end-to-end automation policy, for example, resources managed by System Automation for Multiplatforms, you must manually create a service instance in TBSM and then assign the System Automation Service Template. You also do this if you want to enrich service instances that already exist in a TBSM service tree with information from System Automation events.

A service template consists of rules that can be applied for service instances. A template can be used for more than one instance. If you want to assign the EEZ_SystemAutomationResource template to a service, you can tag the service with the template. Proceed as follows:

- Tag services that are using the EEZ_SystemAutomationResource template to make the defined incoming status rules available to these services:
    1. In the **Service Navigation** portlet, select the service name for which you want to assign the EEZ_SystemAutomationResource System Automation-specific service template.
    2. Select the **Edit Service** tab in the Service Editor to edit the service.
    3. Select the **Templates** tab. You can see the following two lists:
        - *Available Templates*: Displays all templates that you have the permission to assign to the selected service instance.
        - *Selected Templates*: Displays all templates assigned to the service.
    4. To assign the System Automation Service template to a service, select the EEZ_SystemAutomationResource template from the **Available Templates** list. Click the arrow button (>>) to move the template to the **Selected Templates** list.
- Configure the **Identification Field** values for this service. TBSM uses the Identification Fields to map incoming events to a service instance:

1. Select the **Edit Service** tab.
2. Select the **Identification Fields** tab. Here you can find the rules defined in the EEZ_SystemAutomationResource template and the identification field values required to map an event to the selected service instance. The rules contained in the EEZ_SystemAutomationResource template use the AlertKey event attribute as an identifier. By default, the value for each identification field is the value entered in the **Service Name** field.
3. Enter the correct AlertKey attribute value that corresponds to the selected service. The AlertKey value must contain the unique System Automation resource key formatted as CDM SourceToken. The structure is defined like this: EEZResourceKey,DN={DomainName},NN={NodeName},RN={ResourceName},RC={ResourceClass}



Template: **EEZ_SystemAutomationResource**

Rule Name:	Select:	Field Values:
SAExcludedFromAutomationValue:	☐	AlertKey EEZResourceKey,DN={DB2
SAOperationalStateList:	☐	AlertKey EEZResourceKey,DN={DB2
SACompoundStateValue:	☐	AlertKey EEZResourceKey,DN={DB2
SADesiredStateValue:	☐	AlertKey EEZResourceKey,DN={DB2
SAObservedStateValue:	☐	AlertKey EEZResourceKey,DN={DB2
SACompoundState:	☐	AlertKey EEZResourceKey,DN={DB2

To avoid typing errors, open one of the events of the resource and copy and paste the AlertKey value from the event.

Examples for valid AlertKey values:

a) *Resource*: SA MP constituent or fixed resource
Domain: DB2Cluster
Displayed by lssam as: IBM.Application:db2-rs:saxb32c
AlertKey: EEZResourceKey,DN={DB2Cluster},NN={saxb32c},RN={db2-rs},RC={IBM.Application}

b) *Resource*: SA MP move group (Floating Resource)
Domain: DB2Cluster
Displayed by lssam as: IBM.Application:db2-rs
AlertKey: EEZResourceKey,DN={DB2Cluster},NN={ },RN={db2-rs},RC={IBM.Application}

c) *Resource*: SA MP resource group
Domain: DB2Cluster
Displayed by lssam as: IBM.ResourceGroup:DB2
AlertKey: EEZResourceKey,DN={DB2Cluster},NN={ },RN={DB2},RC={IBM.ResourceGroup}

d) *Resource*: SA AM resource group
   Domain: E2E_Domain
   Displayed by the Operations Console as: Online Trading Application
   AlertKey: EEZResourceKey,DN={E2E_Domain},NN={ },RN={Online Trading Application},RC={ResourceGroup}

e) *Resource:* SA AM resource reference
   Domain: E2E_Domain
   Displayed by the Operations Console as: DB2 and the referenced resource is located in the first-level automation domain DB2Cluster
   AlertKey: EEZResourceKey,DN={E2E_Domain},NN={DB2Cluster},RN={DB2},RC={ResourceReference}

   Note: The referenced first-level automation domain must be used as node name of the AlertKey (NN= {FLA-Domain})

f) *Resource*: SA AM choice group
   Domain: E2E_Domain
   Displayed by the Operations Console as: DB2 CG
   AlertKey: EEZResourceKey,DN={E2E_Domain},NN={},RN={DB2 CG},RC=ChoiceGroup}

4. Click the **Additional** tab for the service instance. The tab shows the additional attributes defined for this service instance. The **IBM_Tivoli_System_Automation_Application_Manager_sourceToken** attribute is used for the launch-in-context action that supports launching from TBSM to the SA Operations Console.

5. Overwrite the attribute value for **IBM_Tivoli_System_Automation_Application_Manager_sourceToken** to contain the SA source token that corresponds to this service instance. This value is the same value that has been specified for the AlertKey attributes on the **Identification** tab.

6. Click **Save** to apply your changes.

Whenever a new System Automation state change event is received that matches the specified AlertKey for the service, TBSM now processes the incoming status rules and potentially changes the overall state of the service based on the event severity.

2.2.5 Customizing TBSM views to add information from System Automation

The EEZ_SystemAutomationResource Service Template contains text-based Incoming Status Rules that retrieve the System Automation Observed State and other System Automation specific states of a resource. This information can be used in TBSM Views to enrich service instances with information coming from System Automation.

The following text-based incoming status rules are available:

Rule Name	Description
SAObservedStateValue	Retrieves the SAObservedState field from a resource status change event. **Possible values:** 1. Unknown 2. Online 3. Offline 4. Starting 5. Stopping 6. NotApplicable
SADesiredStateValue	Retrieves the SADesiredState field from a resource status change event. **Possible values:** 1. Online 2. Offline 3. NoChange (i.e. the resource's automation goal cannot be changed by an operator)
SAOperationalStateValue	Retrieve the SAOperationalStateValue field from a resource status change event. A list of Operational State values gives more granular information about the current state of the resource. For a list of possible values, see the `SystemAutomation.baroc` file.
SACompoundStateValue	Retrieve the SACompoundStateValue field from a resource status change event. Compound State indicates whether the resource is working as desired or has encountered an error. **Possible values:** 1. Ok 2. Warning 3. Error 4. Fatal
SAExcludedFromAutomationValue	Retrieve the SAExcludedFromAutomationValue field from a resource status change event. A flag indicates if the resource is excluded from automation (that is, automation is suspended). **Possible values:** 1. NotExcluded 2. Excluded

2.2.5.1 Adding columns for additional System Automation information to a TBSM service tree

You can modify the columns of custom trees displayed in TBSM in the following portlets:

- Service Navigation portlet
- Service Tree portlet

The default Service Navigation portlet has three columns:

- State
- Time
- Events

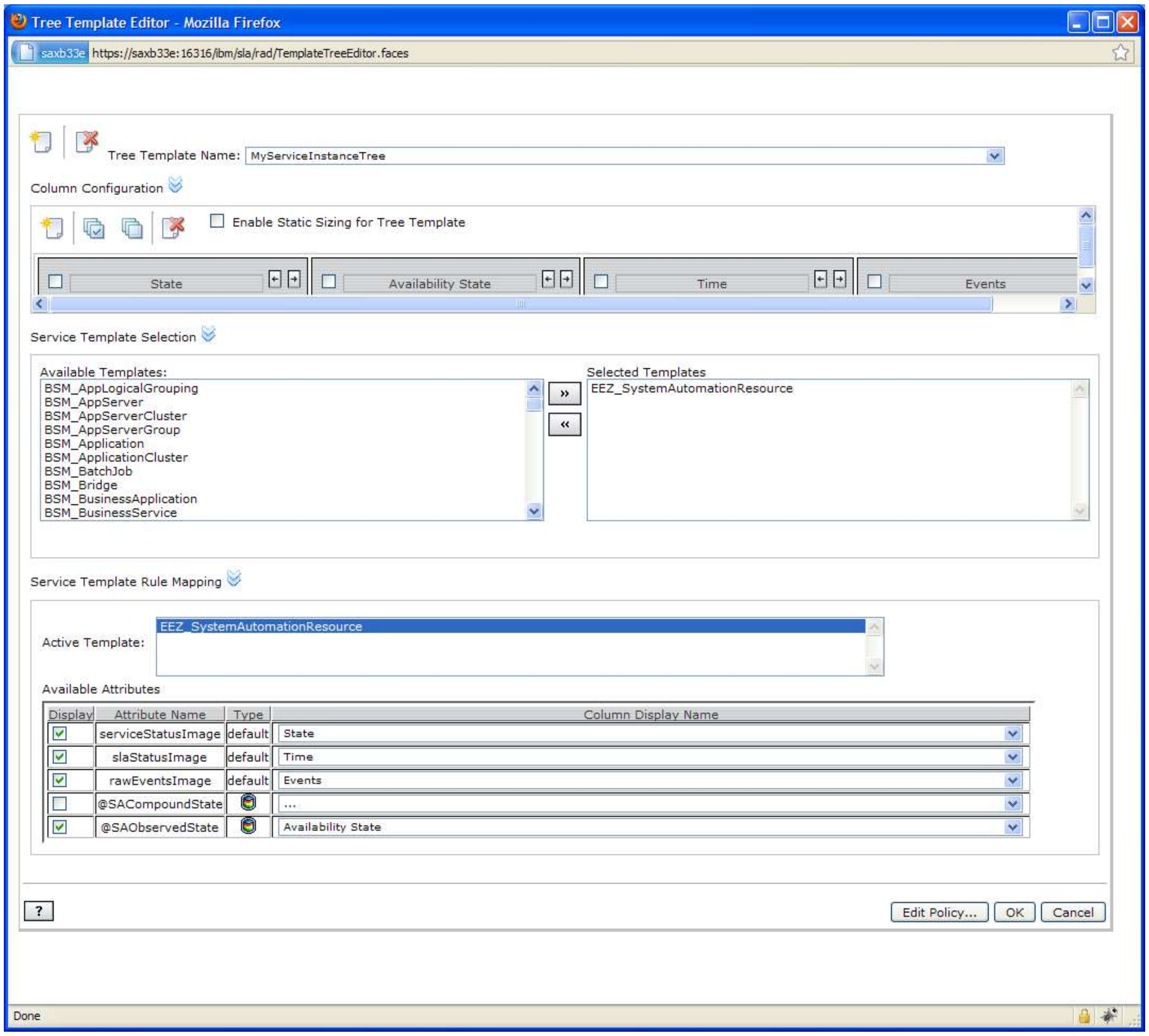
You can modify, delete, and add tree columns with the Tree Template Editor. The Tree Template Editor is available from the Services toolbar in the Service Navigation portlet. For each custom column, use the Tree Template Editor to specify the rule data you want to display in the column.

2.2.5.1.1 Adding columns:

Use the following procedure to add columns for any of the provided text-based incoming status rules defined by the EEZ_SystemAutomationResource template. For example, you can define a column that displays the current Observed State coming from System Automation for each service instance that has the EEZ_SystemAutomationResource template assigned. Complete the following steps:

1. Click the **Tree Template Editor** button in the toolbar of the Service Navigation portlet.
2. Select the tree template you want to modify in the **Tree Template Name** drop-down list.
3. Click the **Add New Tree Column** button in the Column Configuration section.
4. Type the name you want to use in the blank field for the new column, for example, "Availability State".
5. Adjust the column position and width as appropriate
6. In the **Service Template Selection section**, select the EEZ_SystemAutomationResource template.
7. In the **Service Template Rule Mapping**, select the EEZ_SystemAutomationResource template in the Active Template list.
8. For each rule that you want to display in a service tree column, select the **Display** check box and choose a column from the drop-down box to display the output value. In this example, select the Display check box for the attribute @SAObservedStateValue and choose the "Availability State" column from the drop-down box of that row.
9. Click **OK** to save the changes to the tree template.

The following figure shows a screen capture of the tree template editor. A new Availability State column is added showing the System Automation Observed State:


Tree Template Editor - Mozilla Firefox

https://saxb33e:16316/ibm/sla/rad/TemplateTreeEditor.faces

Tree Template Name: MyServiceInstanceTree

Column Configuration

☐ Enable Static Sizing for Tree Template

Columns: State | Availability State | Time | Events

Service Template Selection

Available Templates:
BSM_AppLogicalGrouping
BSM_AppServer
BSM_AppServerCluster
BSM_AppServerGroup
BSM_Application
BSM_ApplicationCluster
BSM_BatchJob
BSM_Bridge
BSM_BusinessApplication
BSM_BusinessService

Selected Templates
EEZ_SystemAutomationResource

Service Template Rule Mapping

Active Template: EEZ_SystemAutomationResource

Available Attributes

Display	Attribute Name	Type	Column Display Name
✓	serviceStatusImage	default	State
✓	slaStatusImage	default	Time
✓	rawEventsImage	default	Events
☐	@SACompoundState		...
✓	@SAObservedState		Availability State

Done

To view the updated Services Tree, refresh the Service Navigation portlet. The new column now shows the output of the incoming status rule that you have selected.

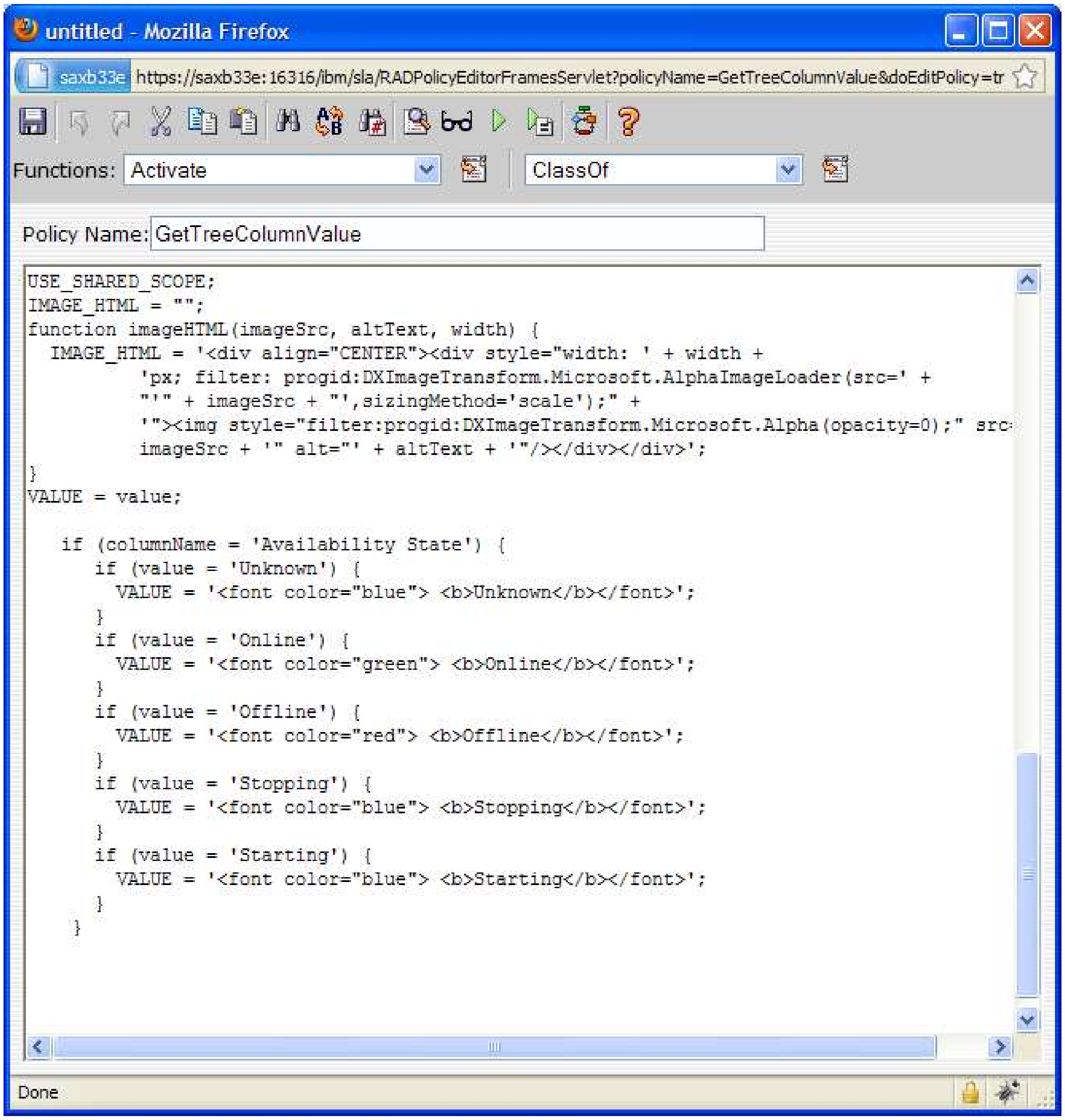
Note: You have to create new resource status change events to update the state information displayed in TBSM. Old events are not processed again.

2.2.5.1.2 Using The TBSM policy editor:

Optionally, you can format column values by using the TBSM policy editor. For example, display the SA Observed State values in different colors. Proceed as follows:

1. Click the **Tree Template Editor** button in the toolbar of the Service Navigation portlet.

2. Choose the tree template you want to modify from the **Tree Template Name** drop-down list.
3. Click on the **Edit Policy** button to open the policy that displays column values. The GetTreeColumnValue policy opens in the policy editor:


untitled - Mozilla Firefox
saxb33e
https://saxb33e:16316/ibm/sla/RADPolicyEditorFramesServlet?policyName=GetTreeColumnValue&doEditPolicy=tr
Functions: Activate
ClassOf
Done

Policy Name: GetTreeColumnValue

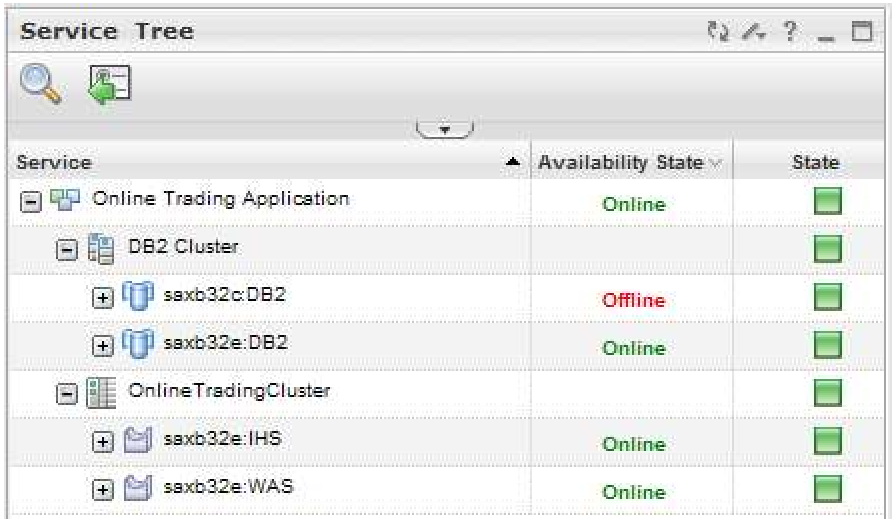
```
USE_SHARED_SCOPE;
IMAGE_HTML = "";
function imageHTML(imageSrc, altText, width) {
  IMAGE_HTML = '<div align="CENTER"><div style="width: ' + width +
          'px; filter: progid:DXImageTransform.Microsoft.AlphaImageLoader(src=' +
          "'" + imageSrc + "',sizingMethod='scale');" +
          '"><img style="filter:progid:DXImageTransform.Microsoft.Alpha(opacity=0);" src=
          imageSrc + '" alt="' + altText + '"/></div></div>';
}
VALUE = value;

   if (columnName = 'Availability State') {
      if (value = 'Unknown') {
        VALUE = '<font color="blue"> <b>Unknown</b></font>';
      }
      if (value = 'Online') {
        VALUE = '<font color="green"> <b>Online</b></font>';
      }
      if (value = 'Offline') {
        VALUE = '<font color="red"> <b>Offline</b></font>';
      }
      if (value = 'Stopping') {
        VALUE = '<font color="blue"> <b>Stopping</b></font>';
      }
      if (value = 'Starting') {
        VALUE = '<font color="blue"> <b>Starting</b></font>';
      }
    }
```

4. Modify the policy. The following code snippet is an example of how to change the color of the text-based output values. In this example, a column named "Availability State" has been defined that shows the output of the SAObservedState Rule. Depending on the value of the observed state, the policy snippet returns the value in a different color:

```
  if (columnName = 'Availability State') {
    if (value = 'Unknown') {
      VALUE = '<font color="blue"> <b>Unknown</b></font>';
    }
    if (value = 'Online') {
      VALUE = '<font color="green"> <b>Online</b></font>';
    }
    if (value = 'Offline') {
      VALUE = '<font color="red"> <b>Offline</b></font>';
    }
    if (value = 'Stopping') {
      VALUE = '<font color="blue"> <b>Stopping</b></font>';
    }
    if (value = 'Starting') {
      VALUE = '<font color="blue"> <b>Starting</b></font>';
    }
}
```

5. Save the modified policy.

The following screen capture shows a Service Tree with a new Availability State column. The System Automation Observed State is displayed in different colors depending on the state:



Service	Availability State	State
Online Trading Application	Online	🟩
DB2 Cluster		🟩
saxb32c:DB2	Offline	🟩
saxb32e:DB2	Online	🟩
OnlineTradingCluster		🟩
saxb32e:IHS	Online	🟩
saxb32e:WAS	Online	🟩

2.2.6 Configuring launch-in-context support

You can define a launch-in-context entry for a TBSM service tree that enables you to launch the SA Operations Console in context of the currently selected service instance.

2.2.6.1 Additional attributes used to supply context information

The launch-in-context action uses the following attributes of a service that must be stored in a service's additional attributes to perform the launch:

IBM_Tivoli_System_Automation_Application_Manager_sourceContactInfo

This attribute contains the launch URL including the host name and the port number of the System Automation Application Manager.

For example:

```
http://saxb45.boeblingen.de.ibm.com:9060/ibm/EEZUIWebClient/
EEZIscUrlBuilderServlet
```

IBM_Tivoli_System_Automation_Application_Manager_sourceToken

This attribute contains the unique source token attribute as defined by the System Automation Application Manager DLA. It represents the resource key identifying the System Automation resource corresponding to this service instance. For example:

```
EEZResourceKey,DN=(FriendlyE2E},NN={},RN={DR_Portal},RC={ResourceGroup}
```

For System Automation Application Manager resources contained in an end-to-end automation policy that are imported using the Discovery Library Toolkit, both attributes are automatically filled with the correct values.

2.2.6.1.1 Specifying launch information for manually created service instances

If the resource is manually created without the Discovery Library Toolkit, for example, for System Automation for Multiplatforms resources, these attributes must be manually specified if launch-in-context to the SA Operations Console is used.

The sourceContactInfo field can be specified in the EEZ_SystemAutomationResource Service Template because it contains generic information that is valid for all System Automation resources. See "Specify SourceContactInfo in Service Template" following section for a description of how to do this.

The sourceToken field must be specified for each service instance that has not been imported by using the Discovery Library Toolkit. See steps 4 and 5 in the "Manually assigning the service template to a service instance" section for a description how to do this step.

Specify SourceContactInfo in Service Template

The additional IBM_Tivoli_System_Automation_Application_Manager_sourceContactInfo attribute contains the URL of the System Automation Application Manager host to which the launch-in-context action will connect. If you want to use launch-in-context for services that have been manually tagged with the EEZ_SystemAutomationResource template, you must manually adapt the URL in the Service Template. This step is not required if you have services that are only imported by using the Discovery Library Toolkit.

Complete the following steps to adapt the EEZ_SystemAutomationResource template for launch-in-context:

1. Select **Templates** from the Service Navigation drop-down menu.

2. Select the **EEZ_SystemAutomationResource** template in the Service Navigation portlet
3. Click the **Edit Template** tab in the **Service Editor** to edit the template.
4. From the **Edit Template** tab, click the **Additional** tab.
5. The **IBM_Tivoli_System_Automation_Application_Manager_sourceContactInfo parameter** has the default value: http://<SA_lSC_Hostname>:9060/ibm/EEZUIWebClient/EEZIscUrlBuilderServlet.
   Replace <SA_lSC_Hostname> with the correct host name of the system running the Integrated Solutions Console of System Automation Application Manager. Also change the port number if required.
6. Click the **Save** button in the toolbar to apply your changes

2.2.6.2 Defining launch-in-context action

This section describes how to define a new launch-in-context (LiC) action for a custom service view in TBSM, which will be displayed in the context menu of a service instance.

You must first create a new view definition. By using the View Definitions feature of TBSM, you can change the visual content that is displayed in your Service Editor/Viewer portlet and change the values that are displayed in the visual elements. You may also define Actions. *Actions* are the options available to you as the result of clicking, double-clicking, or right-clicking a service in the Service Editor.

You cannot modify the predefined View Definitions, therefore you have to create a new View Definition to define a new Action:

1. Select **Services** from the **Service Navigation** drop-down menu.
2. Click a service in the Service Navigation portlet. The default view definition for the service model opens in the Service Editor/Viewer.
3. In the **View Service** tab, select the view definition you want to work with from the View Definition drop-down list.
4. Click the **Edit View Definition** button. The Edit View Definition window opens
5. To create a view definition, click **Save as New**. The Save As window opens.
6. Type the name of your view definition in the **Save as New** field and click **OK**.

Next, you must define a new LiC-action for the new view definition:

1. In the View Definition drop-down list, select your View Definition and click the **Edit View Definition** button to reopen the Edit View Definition window.
2. Click the **Actions** tab in the Edit View Definition window.
3. To add a new action, click the **Edit** button next to the **Edit Action** selection list. The Edit Canvas Action window opens.

4. Define a new Action by using the following parameters:

Attribute	**Value**
Action Name	`ShowSAResource`
Action Display Name	`Show Service in SA Operations Console`
Action Description	`Launch the service in the Tivoli System Automation Operations Console`
Action URL	`__IBM_Tivoli_System_Automation_Application_Manager_sourceContactInfo__?SourceToken=__URLEncode(IBM_Tivoli_System_Automation_Application_Manager_sourceToken)__`
Action Frame	`SystemAutomation`

5. Click **OK**.

6. Add the new action to the context menu. You have two options:

   a) If you want the launch menu entry to be visible only for services that have the EEZ_SystemAutomationResource template assigned as primary template, select the EEZ_SystemAutomationResource template from the Service Template drop-down list.

   b) If you want the launch menu entry to be always visible for all kinds of services, click the **Set Defaults** button.

   Note: If you have assigned the EEZ_SystemAutomationResource to services that have been tagged with a different primary template, you have to use option b) to use the launch-in-context action for these services.

7. To add the new action as a right-click action, click the **New** button to add a new row to the **Right Click Menu Options** action list. A blank row is added to the Actions list.

8. Select the new action from the drop-down list.

9. Click **Apply** to save the action.

10. Click **OK** to close the Edit View Definition window.

2.2.6.3 Exporting System Automation specific additional attributes for use in LiC Actions

By default, only certain attributes are available for actions. To use the IBM_Tivoli_System_Automation_Application_Manager_sourceContactInfo and

IBM_Tivoli_System_Automation_Application_Manager_sourceToken attributes in the LiC action, you must edit the xml file for the view definition and add a fieldToPassToModelExpr tag for each attribute that is used in the action.

The `$TBSM_DATA_SERVER_HOME/av/xmlconfig` directory contains various xml configuration files, including the files that configure view definitions. View definition xml files have the format:

ViewDefinition_ViewDefName.xml

where ViewDefName is the name of the view definition in the TBSM console. For example, the name of the file for a custom view definition called MyView is:

ViewDefinition_MyView.xml

Add the following elements to the view definition xml for which you want to enable the LiC action:

```
<fieldToPassToModelExpr
modelField="IBM_Tivoli_System_Automation_Application_Manager_sourceContactInfo">IBM_Tivoli_S
ystem_Automation_Application_Manager_sourceContactInfo</fieldToPassToModelExpr>
<fieldToPassToModelExpr
modelField="IBM_Tivoli_System_Automation_Application_Manager_sourceToken">IBM_Tivoli_System_
Automation_Application_Manager_sourceToken</fieldToPassToModelExpr>
```

This step will make the two attributes available for use in the LiC action.

Note: You must restart the Tivoli Integrated Portal Server to activate this change in TBSM.

2.2.6.4 Configuring launch-in-context to launch from System Automation to TBSM

You can set up launch-in-context support to launch from a resource displayed in the SA Operations Console to the TBSM Web GUI. This launch-in-context support enables users to launch the Service Availability Workspace of the Tivoli Integrated Portal and automatically select the corresponding service instance in the TBSM Service Viewer with a single mouse click.

When TBSM launch-in-context support is configured, a hyperlink becomes available in the System Automation operations console on the General page of resources contained in an end-to-end automation domain. This hyperlink allows users to navigate to the corresponding service instance in the TBSM Service Viewer with a single mouse click.

Note: The TBSM launch-in-context support is available only for resources that have been imported into TBSM by using the Discovery Library Toolkit.

Perform the following steps to configure TBSM launch-in-context support for the SA Operations console:

1. Log in to ISC on the system where System Automation Application Manager is installed.
2. Navigate to **Tivoli System Automation > Settings > TBSM Launch-in-Context Configuration**.
3. In the fields on the page that is displayed, do these steps:

Enable launch-in-context support for TBSM

Select to enable launch-in-context support.

Server name used to connect to Tivoli Integrated Portal

Specify the name of the server where the Tivoli Integrated Portal that hosts the TBSM Web GUI runs.

Port number used to connect to Tivoli Integrated Portal

Specify the secure port number of the server where the Tivoli Integrated Portal runs. The default port number is 16316.

4. Click **OK** to save the configuration.

Note: If the operations console is displayed while the TBSM launch-in-context configuration is changed, select **Menu -> Refresh all** to enable the changed settings in the current instance of the Operations Console.