

Course Guide

IBM Jazz for Service Management 1.1 Dashboarding Made Simple

Course code TN700 ERC 1.0



December 2016 edition

NOTICES

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

TRADEMARKS

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

© Copyright International Business Machines Corporation 2016.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this course	xiii
About the student	xiii
Learning objectives	xiv
Course descriptionxv
Unit 1 Dashboard design fundamentals, DASH overview, and strategy	1-1
Objectives	1-2
Lesson 1 Dashboard design overview	1-3
High-level dashboard design	1-4
Dashboard design principles	1-5
Identifying the target personas	1-6
Persona interviews	1-7
Creating prototypes of dashboard pages	1-9
Example: Operational dashboard	1-10
Developing dashboard pages	1-11
Evaluating dashboard designs	1-12
Lesson 2 Visualization services architecture	1-13
Jazz for Service Management components	1-14
Visualization services overview	1-15
Jazz for Service Management components	1-15
Data visualization with linked data	1-16
Visualization services architecture diagram	1-17
Jazz for Service Management installation options	1-18
Providers, data sets, and data sources	1-19
Connections: Widget data pipes	1-20
Creating connections	1-21
Configuring a connection	1-22
Connection authentication	1-23
Exercises	1-24
Lesson 3 Creating dashboard pages	1-25
Page layout styles	1-26
Scalar and list widgets	1-27
Mobile widget overview	1-28
Mobile support	1-29
Creating a dashboard page	1-30
Dashboard workspace elements	1-31
Adding widgets to a dashboard page	1-32
Customizing widgets	1-33
Adding widgets to a dashboard page	1-34
Customizing widget visualization settings	1-35

Selecting a widget dataset	1-36
Connecting dashboard pages with events and wires	1-37
Completed dashboard example	1-38
Event transformations	1-39
Events and wires examples	1-40
Organizing dashboard pages with views and profiles	1-41
Managing dashboards	1-42
Creating authorization roles	1-43
Managing views	1-44
Automatically opening pages in a view	1-45
Creating console preference profiles	1-46
Console preference profile view, task, and navigation bars	1-47
Setting profile authorization and the default view	1-48
Exercises	1-49
Lesson 4 Local web content and dashboard server customizations	1-50
Hosting local web content	1-51
Adding files to the myBox web applications	1-52
Changing login screen parameters	1-53
Lesson 5 Visualization services strategy	1-54
Strategy: common data access, common visualizations	1-55
Strategy: Blaze technology for stand-alone products	1-56
Integrating stand-alone console content	1-57
Adding a console integration	1-58
Completing the console integration	1-59
Strategy: Use DASH to integrate IBM and non-IBM application data	1-60
Exercises	1-61
Summary	1-62

Unit 2 Tivoli Directory Integrator and DASH basics.....	2-1
Objectives	2-2
Lesson 1 Tivoli Directory Integrator overview and architecture	2-3
Tivoli Directory Integrator overview	2-4
Tivoli Directory Integrator Architecture	2-5
Tivoli Directory Integrator and DASH	2-6
Tivoli Directory Integrator assembly lines	2-7
Data connectors	2-8
Connector modes	2-9
Connector data: Entries and attributes	2-10
Assembly line data flow	2-11
Building assembly lines	2-12
Runtime server	2-13
Tivoli Directory Integrator - DASH sites for reference	2-14
Lesson 2 Using the Configuration Editor	2-15
Starting the configuration editor	2-16
Configuration editor interface	2-17

Creating a project	2-18
Configuration Editor debugger	2-19
Breakpoints	2-20
Debugger watch list	2-21
Lesson 3 Creating an AssemblyLine	2-22
Naming DASH Tivoli Directory Integrator projects	2-23
DASH AssemblyLine naming standards	2-24
Adding an assembly line	2-25
Assembly line data flow in the Configuration Editor	2-26
Lesson 4 Creating a UI data provider data set from a database	2-27
Example: Show sales database data in a chart widget	2-28
UI data provider creation tasks	2-29
Creating a DB2 database connector	2-30
Configuring a DB2 database connector	2-31
Verifying the database connection	2-32
Previewing data	2-33
Mapping data source columns to AssemblyLine output	2-34
Reviewing output mappings	2-35
Deploying the AssemblyLine to the runtime server	2-36
Using the data set with a dashboard widget	2-37
Correspondence of data set values and work attributes	2-38
Configuring the column chart	2-39
Completed DASH widget	2-40
Instructor demonstration	2-41
Student exercises	2-42
Lesson 5 Creating a Tivoli Directory Integrator-based data set from a CSV file	2-43
Example: Creating data sets from unstructured data	2-44
CSV file-based data set creation tasks	2-45
Adding an assembly line to an existing project	2-46
Adding a file connector to the assembly line	2-47
Selecting a File Connector object	2-48
Configuring the file connector data source	2-49
Adding a file parser to the connector	2-50
Verifying the file connector operation	2-51
Mapping file data to the input map	2-52
Modifying work attribute output values	2-53
Deploy the assembly line to the runtime server	2-54
Clearing the Tivoli Directory Integrator cache in the DASH server	2-55
Using the file connector data set with a widget	2-56
Completed chart with file connector data set	2-57
Instructor demonstration	2-58
Student exercises	2-59
Unit summary	2-60
Unit 3 Tivoli Directory Integrator and DASH advanced topics	3-1
Objectives	3-2
Lesson 1 Expanding assembly line functions with helper assembly lines	3-3
Helper assembly lines	3-4

Importing assembly lines templates	3-5
Selecting the assembly lines import scope	3-6
Project view after importing the configuration file	3-7
Lesson 2 Creating a parameter helper assembly line	3-8
Example: Add location parameters to car sales data	3-9
Adding parameters to a data set	3-10
Updated database	3-11
Configuring the primary assembly lines	3-12
Configuring the database connector	3-13
Creating an SQL select statement in the database connector	3-14
Using a script for the SQL select configuration	3-15
SQL select after saving script	3-16
Adding a helper assembly lines to a project	3-17
Modifying an assembly lines template	3-18
Configure the DefineParameters script component	3-19
Modifying the DefineParameters template script	3-20
Final Script	3-21
Modifying the database connection to use parameter selections	3-22
SQL Select configuration	3-23
Using the parameterized UI data provider in a dashboard	3-24
Configuring the hotspot widgets	3-25
Configuring the table widget	3-26
Testing the updated Sales by Location page	3-27
Instructor demonstration	3-28
Student exercises	3-29
Lesson 3 Creating a status column helper assembly line	3-30
Adding data set columns to show status in a data set	3-31
Columns assembly lines overview	3-32
Copying the _columns template	3-33
Renaming the new assembly lines	3-34
Defining each column in the helper assembly line	3-35
The StatusUtility script	3-36
Setting the StatusUtility script scope	3-37
Evaluating status thresholds	3-38
Testing the _column helper assembly lines	3-39
Instructor demonstration	3-40
Student exercises	3-41
Lesson 4 Creating a tasks helper assembly line	3-42
Adding task menus to a data set	3-43
How to create a _tasks helper assembly line	3-44
Creating the task helper assembly line	3-45
Renaming the helper assembly line	3-46
Reviewing the sample tasks script	3-47
Looking up an internal page ID	3-48
Modifying the tasks script	3-49
Testing the _tasks helper assembly lines	3-50
Exercises	3-51
Lesson 5 Creating a run-once assembly line	3-52

Run Once assembly line	3-53
Typical assembly line review	3-54
Run-once assembly line review	3-55
Example: Parsing a file and extracting metrics	3-56
Using the run-once assembly line template	3-57
DS_Run_Once assembly line processing flow review	3-58
Modifying the template data connector	3-59
Modifying the file connector	3-60
Configuring the CSV file parser	3-61
Testing the file connector	3-62
Mapping input values	3-63
Lets Use a Simple Example	3-64
ProcessEntry script component logic review	3-65
Clear work script component review	3-67
Configuring the set reply component	3-68
Testing the run-once assembly line	3-69
Exercises	3-70
Unit summary	3-71

Unit 4 DASH and Netcool/Impact integration	4-1
Objectives	4-2
Lesson 1 Netcool/Impact overview	4-3
Netcool/Impact overview	4-4
High level architecture	4-6
Primary features	4-9
Netcool/Impact UI data provider tools	4-10
Client use cases	4-11
Dashboard visualization	4-12
The Netcool/Impact console	4-13
Lesson 2 Creating data sets from Netcool/Impact data types	4-14
Creating an Impact data type	4-15
Creating a data model	4-16
Configuring the data model data source	4-17
Creating a data type	4-18
Configuring a key field for the data type	4-20
Using an Impact data type UI data provider in DASH	4-21
Lesson 3 Extending data set content with data type custom output parameters	4-22
Adding status data to a data type UI data provider	4-23
Creating custom output parameters in the data type	4-24
Custom data type output parameter results	4-25
Creating custom output parameters with a list widget	4-26
Lesson 4 Creating Netcool/Impact data sets with policy scripts	4-27
Creating a UI data provider with an Impact policy	4-28
Creating an Impact policy	4-29
Example: Creating a SQL query policy	4-31
Creating policy output parameters	4-32
Using an Impact policy UI data provider in DASH	4-33
Mandatory parameter: executePolicy	4-34

Lesson 5 Creating data sets from policy script variables	4-35
Using policy variables as UI data provider output parameters	4-36
Supported output parameter types	4-37
Output parameter notes	4-38
Creating the trouble ticket data type	4-39
Creating the policy	4-40
Creating the output parameters	4-41
Using a variable policy data set in a DASH widget	4-42
Exercises	4-43
Unit 5 Using the Jazz for Service Management web widget with business dashboards	5-1
Objectives	5-2
Lesson 1 Web widget overview	5-3
Dashboards and web-based content	5-4
Web widget architecture	5-5
Web widget configuration	5-6
Widget events overview	5-7
Modifying the widget event publish and subscribe configuration	5-8
Event wires overview	5-9
Event transformations	5-10
DASH tools overview	5-11
Adding Event Spy widget to a page	5-12
Viewing events in the debugger console	5-13
NodeClickedOn widget overview	5-14
Instructor demonstration	5-15
Student exercises	5-16
Lesson 2 Web widget event parameter substitution	5-17
Parameter substitution in the web widget home page	5-18
Parameter Substitution Example	5-19
Serving web content with myBox web archive file	5-20
Managing myBox content	5-21
Examples: Serving files with the myBox web archive	5-22
Lesson 3 Web widget parameter substitution with JavaServer pages	5-23
Using web widgets with JSP scripts	5-24
Configuring a web widget to use a JSP file	5-25
Exercises	5-26
Lesson 4 Using web widgets to show non-IBM application consoles	5-27
Adding application consoles to dashboard pages	5-28
Web widget configuration with application consoles	5-29
Lesson 5 Using web widgets to show non-IBM application consoles	5-30
Hotspot widget overview	5-31
Configuring hotspot widgets to send events	5-32
Lesson 6 Using web widgets as a hotspot widget target	5-33
Smart text widget overview	5-34
Inserting data set tags	5-35
Adding tables	5-36
Formatting smart text content	5-37

Instructor demonstration	5-38
Exercise	5-39
Summary	5-40
Unit 6 Integrating Netcool/OMNibus with DASH.....	6-1
Objectives	6-2
Lesson 1 OMNibus overview	6-3
Deployment architecture	6-4
Getting started	6-6
Lesson 2 WebGUI UI data provider data sets	6-7
Provider and data sources overview	6-8
Events data set: Events data source	6-9
Filter Summary data set: Aggregated data data source	6-10
Event Summary data set: Aggregated data data source	6-11
Event Grouping data set: Aggregated data data source	6-12
Event Fields data set: Event information data source	6-13
Event Details data set: Event information data source	6-14
Journal Entries data set: Event information data source	6-15
Lesson 3 Using DASH widgets with Netcool/OMNibus event data	6-16
Populating DASH widgets from OMNibus Web GUI provider	6-17
Events that match a specific condition	6-18
Maximum severity of events	6-19
Set of events in a gauge or volume bar	6-20
Set of fields for selected events	6-21
Field values, details, or journal entries for a single event	6-22
Events grouped by multiple fields	6-23
Lesson 4 Transient filters and dynamic event lists	6-24
Creating filtered event lists with context events	6-25
OMNibus transient filters	6-26
Using event viewers in a web widget	6-27
Opening event viewers in a web widget	6-28
Event viewer URL query strings	6-29
Example: Showing events related to a table selection	6-30
Lesson 5 Troubleshooting the WebGUI UI data provider	6-31
Checking Web GUI data provider registration	6-32
Checking the DASH connection with the ObjectServer	6-33
Diagnosing problems	6-34
Exercises	6-35
Summary	6-36

Unit 7 Integrating IBM Tivoli Monitoring and DASH.....	7-1
Objectives	7-2
Lesson 1 IBM Tivoli Monitoring overview	7-3
Enterprise monitoring	7-4
IBM Tivoli Monitoring components	7-5
ITM user interface strategy	7-6
Tivoli Enterprise Monitoring Agents	7-7
Portal Java client	7-8
Key Requirements for UI data provider data access	7-9
System Status and Health dashboards	7-10
Attributes	7-11
Query	7-12
Groups	7-13
Lesson 2 Connecting Tivoli Monitoring for dashboards	7-14
IBM Tivoli Monitoring UI data provider	7-15
IBM Tivoli Monitoring dashboard data provider (continued)	7-16
Creating connections	7-17
Creating connections, continued	7-18
Selecting a connection	7-19
Verifying that the dashboard data provider is enabled	7-20
Verifying Tivoli Monitoring dashboard	7-21
Carousel view	7-22
Starting the Tivoli Enterprise Portal client	7-23
Lesson 3 Building dashboards	7-24
Editing a Widget and Specifying Data Sets	7-25
Examples of ITM Data Providers and Data Sources	7-26
Creating a dashboard page	7-27
Selecting the data set	7-28
Selecting the attribute	7-29
Finishing the settings	7-30
Memory Monitor dashboard	7-31
Lesson 4 Troubleshooting	7-32
Locating Attributes and Datasets	7-33
Data Provider logs and traces	7-34
DataProvider logs contd.	7-35
DataProvider logs contd.	7-36
Exercises	7-37
Instructor demonstration	7-38
Summary	7-39

Unit 8 Integrating with Tivoli Business Service Manager	8-1
Objectives	8-2
Lesson 1 Tivoli Business Service Manager overview	8-3
Business service management definitions	8-4
Business service example: Online banking	8-5
Defining business services with service models	8-6
Service status and key performance indicators	8-7
Service template	8-8
Business services and service status	8
Incoming status rules	8-10
Dependency rules	8-11
Numerical modeling	8-12
Showing key performance indicators	8-13
Lesson 2 Tivoli Business Service Manager data provider architecture	8-14
Tivoli Business Service Manager data provider architecture diagram	8-15
UI data provider overview	8-16
Data sources	8-17
TBSM Services data source	8-18
TBSM Templates data source	8-19
TBSM Primary Templates data source	8-20
TBSM Topology data source	8-21
TBSM Tree Template data source	8-22
Dynamic updating	8-23
Dynamic updating (continued)	8-24
Context menus	8-25
Lesson 3 Using data widgets with Tivoli Business Service Manager data sets	8-26
Correlating template rules and template datasets	8-27
Example: Analog gauge with TBSM Primary Templates data source	8-28
Selecting the analog gauge dataset column value	8-29
Optional visualization settings: Labels	8-30
Optional visualization settings: Values	8-31
Analog gauge: Selecting a service	8-32
Analog gauge: Enhanced parameter selector	8-33
Example: Tree table and topology data source	8-34
Tree table widget: Optional settings	8-35
Optional settings: Save user settings	8-36
Completed tree table widget	8-37
Lesson 4 Troubleshooting	8-38
Troubleshooting: Is it TBSM?	8-39
Verify the DASH version	8-40
Incorrect service name	8-41
Known problems and limitations	8-42
Known problems and limitations (continued)	8-43
No services being displayed	8-44
Troubleshooting: Unexpected status value	8-45
Best Practice: Use of generic provider ID for connections	8-46
Best Practice: Synchronizing DASH status with TBSM status	8-47
Reference Materials	8-48

Exercises	8-49
Summary	8-50
Unit 9 Networks for Operations Insight dashboards	9-1
Objectives	9-2
Lesson 1 Networks for Operations Insights overview	9-3
Netcool Operations Insight base features	9-4
Networks for Operations Insight	9-5
Netcool Operations Insight 1.4 with Network option	9-6
Additional optional components	9-7
Base solution components	9-8
Optional network management components	9-8
Networks for Operations Insight installation considerations	9-10
Lesson 2 Network health dashboards review	9-11
Overview	9-12
Network Health Dashboard (NHD)	9-13
Network Views	9-14
Network Availability (1)	9-15
Network Availability (2)	9-17
Polling Metrics	9-18
Configuration and Event Timeline (1)	9-20
Configuration and Event Timeline (2)	9-21
Instructor demonstration	9-23
Unit 10 Exporting and importing customizations	10-1
Lesson 1 Exporting and importing dashboard customizations	10-2
Objectives	10-3
Export and import scope	10-4
Exporting dashboard customizations	10-5
Export wizard example	10-6
Export wizard example (continued)	10-7
Export command line examples	10-8
Import command line examples	10-9
Lesson 2 Workshop summary and dashboard review	10-10
Successful PoC review	10-11
General tips about PoC data	10-12
Are you Ready for a POC?	10-13
Resource when you have questions	10-14
Student dashboard presentations and questions	10-15
Exercises	10-16
Student demonstrations	10-17
Summary	10-18

About this course

This workshop will teach students how to design, develop, and deploy business dashboards. The student will learn a best practice methodology for engaging customers to learn their business dashboard requirements, designing prototype dashboards, tools and techniques for mining data from a variety of data sources. The data sources include IBM applications, such as Netcool Operations Insight, IBM Tivoli Monitoring, IBM Tivoli Business Services Manager, Tivoli Application Dependency Discovery Manager, and Netcool Network Management. The student will also learn tools and techniques to mine business data from non-IBM sources, such as databases, Internet feeds, and legacy application consoles. You will learn how to use IBM Netcool/Impact and Tivoli Directory Integrator to create DASH data providers that extract and enhance data from back-end data sources. You will also learn how to federate and include supported application consoles, such as Smart Cloud Application Performance Management UI.

The lab environment for this course uses RedHat Enterprise Linux and SUSE Linux Enterprise Server virtual servers.

For information about other related Cloud and Smarter Infrastructure courses, visit the education training paths website:

ibm.com/software/

Details	
Delivery method	Classroom or instructor-led online (ILO) or self-paced (SPVC)
Course level	ERC 1.0 This course is a new course.
Product and version	Jazz for Service Management 1.1.2.0
Duration	4 days
Skill level	Advanced

About the student

This course is designed for support, technical sales, services, and all internals.

Before taking this course, make sure that you have basic understanding of the Jazz for Service Management dashboard server tools, have basic understanding of database design and operation,

including concepts such as tables, views, and schemas, and have a basic understanding of Internet design and terminologies, including HTTP, JSP, URI/URL, and REST.

Learning objectives

After you complete this course, you can perform the following tasks:

Define and develop dashboard design documentation

Use Tivoli Directory Integrator to extract and transform back-end data and present the data to DASH widgets through custom UI data providers

Use advanced Tivoli Directory Integrator data mining techniques to enhance the custom UI data provider function, including generating status data, parameter data, and right-click task menus

Use Netcool/Impact to create custom data providers with database data sources

Use web widgets to add custom web-based data to dashboards, and use widget debugging tools to troubleshoot dashboard design problems

Integrate Netcool/OMNIbus event data with DASH dashboard widgets

Integrate IBM Monitoring agent data with DASH dashboard widgets

Integrate IBM Tivoli Business Service Manager service model data with DASH dashboard widgets

Explain the dashboard integration points for Networks for Operations Insight

Course description

The course contains the following units:

1. [Dashboard design fundamentals, DASH overview, and strategy](#)

In this unit, you review the high-level design of the Jazz for Service Management Dashboard Application Service Hub (DASH) server and how dashboards are created and used. You learn how the basic concepts that are used when designing dashboards. This material is independent of any dashboard technology. You review the basic steps involved with building dashboards, and how to customize the console login page. Finally, you review the current visualization services design strategy and how the DASH server supports that strategy.

In the exercises in this unit, you develop and review basic dashboard development skills. You begin the process of designing a set of connected dashboard pages for a customer. You use the design to develop and present the dashboards at the end of the workshop when you develop skills in mining and showing data from many different sources. Next, you start the DASH server components and review the basic elements of the DASH server console. You then add custom web content to the myBox web server object that is installed in the DASH server. The myBox web server is used to service custom graphics, web pages, and JSP pages.

2. [Tivoli Directory Integrator and DASH basics](#)

This unit reviews the Tivoli Directory Integrator high-level architecture and how it is used to extract and transform data from a variety of back-end data sources. You learn how to use the Tivoli Directory Integrator configuration editor to create AssemblyLines that function as custom UI data providers.

In these exercises, you learn the basic skills to use Tivoli Directory Integrator to extract data from simple data sources (a database, a CSV file). You create a UI data provider to show database and CSV file data in a DASH widget. You also learn how to create and debug Tivoli Directory Integrator projects and assembly lines with the Configuration Editor (CE).

3. [Tivoli Directory Integrator and DASH advanced topics](#)

This unit reviews advanced techniques for using Tivoli Directory Integrator to extract and transform data from a variety of back-end data sources and then providing the enhanced data to the DASH server. The student will learn how to reuse common Tivoli Directory Integrator assembly lines to speed up the process of data provider development. You learn how to use helper AssemblyLines to add status data, parameter data, and right-click tasks to data set data. You learn how to create an AssemblyLine that can calculate a metric value from a data source, and how to use a caching connector to support dashboard presentations when a data source is not available.

In these exercises, you learn advanced techniques to retrieve data and create UI data providers with Tivoli Directory Integrator. You learn how to enhance UI data providers with helper assembly lines that provide parameter selections and status evaluation. You learn how to save and use cached data to provide offline demonstrations and how to evaluate and present metric data from a data source.

4. [DASH and Netcool/Impact integration](#)

This unit reviews how to use Impact to transform data from a variety of back-end data sources and then provide the enhanced data to the DASH server as data provider data sets.

The exercises in this unit provide hands-on demonstrations of how to use Netcool/Impact tools to serve data to DASH dashboard widgets with custom UI data provider data sets.

5. [Using the Jazz for Service Management web widget with business dashboards](#)

The DASH Web widget is very versatile for serving web-based content in dashboard pages. In this unit, you learn how to use the built-in myBox web application container to host images, jsp files, and html on the DASH server. You learn how to use hotspot events and command substitution to dynamically modify widget data content, based on the selection of hotspots and other widgets.

The exercises in this unit show several techniques to use web widgets to show business dashboard data. You first learn how to use debugging tools to see the parameters that are available in published widget events. You then learn how to pass event parameters to the web widget URL address. Next, you learn how to pass multiple parameters in a web widget URL to a JavaServer Page (JSP) file. You use this technique to create a web widget that shows dynamic text. Finally, you learn how to use hotspot widgets to pass data to web widgets and create interactive dashboard pages.

6. [Developing a dashboard proof of concept](#)

In this unit, you learn how to gather customer dashboard design criteria, and how to use that information to develop a customer proof-of-concept. You review the effect of scope-creep on a dashboard design and review design development resources that are available before and during a proof-of-concept delivery.

In this unit, you continue to develop the dashboard design documents that you started in Exercise 1, “Creating dashboard prototype diagrams,” on page 1.

7. [Integrating Netcool/OMNibus with DASH](#)

In this unit, you learn how to integrate Netcool/OMNibus event data with DASH dashboards. You learn how to use Netcool/OMNibus data filters to use in DASH data sets. You also learn how to troubleshoot data communication problems between the DASH server and the Netcool/OMNibus UI data provider.

Netcool/OMNibus provides data feeds of event-based and metric-based data from the Web GUI server. In these exercises, you use several DASH widgets to add Netcool/OMNibus event data to a DASH dashboard.

8. [Integrating IBM Tivoli Monitoring and DASH](#)

In this unit, you learn the IBM Monitoring dashboard integration points. You learn how to retrieve monitoring agent data from the UI data provider and use that data in several types of dashboard widgets. You learn how the system status and health dashboards can be implemented to provide quick operational dashboards that provide an enterprise-wide view of your monitoring environment.

This lab exercise demonstrates how to create IBM Tivoli Monitoring dashboards in the DASH server. The custom dashboard that you create uses the Tivoli Enterprise Portal UI data provider. The data provider retrieves IBM Tivoli Monitoring agent data and display the selected attributes in a dashboard page.

9. [Integrating with Tivoli Business Service Manager](#)

In this unit, you learn how to use the IBM Tivoli Business Service Manager UI data provider to serve business service data to dashboard pages. You learn how to configure different widgets with several of the available data sets, including topology data. You learn how to create multiple-page dashboards that show high-level data views and can show context-driven detail information in secondary dashboard pages. Finally, you learn how to configure right-click menu data to data sets. The right-click menus are available to compatible widget types, such as list, table, or tree table widgets.

In these exercises, you use the DASH server tools to show Tivoli Business Service Manager business service data in two connected business dashboard pages. You also learn how to add and use widget context tasks with DASH dashboard widgets.

10. [Networks for Operations Insight dashboards](#)

In this unit, you review the integration points between Networks for Operation Insight and DASH. You review, at a high level, the Networks for Operation Insight architecture and the ways in which network event data can be used with dashboards.

There are no student exercises for this chapter.

11. [Exporting and importing customizations](#)

In this unit, you review the dashboard design process and summarize the responsibilities and tips for delivering a successful customer proof-of-concept.

In these exercises, workshop students export their DASH customizations and finalize their dashboard design and development and present their dashboards to the workshop.

Unit 1 Dashboard design fundamentals, DASH overview, and strategy

IBM Training



Unit 1 Dashboard design fundamentals, DASH overview, and strategy

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this unit, you review the high-level design of the Jazz for Service Management Dashboard Application Service Hub (DASH) server and how dashboards are created and used. You learn how the basic concepts that are used when designing dashboards. This material is independent of any dashboard technology. You review the basic steps involved with building dashboards, and how to customize the console login page. Finally, you review the current visualization services design strategy and how the DASH server supports that strategy.

Objectives

After completing this unit, you should be able to perform the following tasks:

- Describe the steps that are required to gather dashboard design criteria
- Design high-level dashboard page diagrams based on customer requirements
- Create connection documents to remote UI data providers
- Build interconnected dashboard pages and manage user access to pages, views, and console preference profiles
- Configure locally served web content, such as graphic images, HTML, and JSP pages that can be used in dashboard pages

Lesson 1 Dashboard design overview

IBM Training



Lesson 1 Dashboard design overview

This course is geared toward making dashboard designs. These designs can be made by you for your company or can be something that you create for another company.

High-level dashboard design

- The initial design process is independent of dashboard technology
 - Identify the target personas
 - Interview persona representatives
 - Prototype
 - Develop
 - Evaluate



High-level dashboard design

Determine what it is that your community would like to see in the dashboards. Determine the layout and flow of what is about to be designed.

Dashboard design principles

- Focus on target audience
- Avoid mixing dashboard types, such as technical versus executive
- Avoid clutter
- Mine and surface the needed data

Dashboard design principles

These general principles will assist you to create good designs. A big part of this effort is how you get the data from the sources into the widgets.

Identifying the target personas



- UI service consumer
 - Users or groups that use operational support services (OSS) or business support services (BSS)
 - Examples include business managers, analysts, subject matter experts (SME), Customer Support personnel, and Security personnel
- UI service provider
 - Users or groups that provide and maintain OSS or BSS
 - Examples include operations managers, line-of-business (LOB) managers, and service providers
- UI service creators
 - Users or groups that design and create UI services for consumers and providers
 - Users or groups that create, maintain, and administer UI data sources

Persona interviews

- Identify what the user or group needs to see in a dashboard
 - What do they need to do their work?
- Identify the data that is required in each dashboard
- Identify the sources of the data



Persona interviews

Attempt to itemize the specific users that you want to provide dashboard(s) for

Identify their wants and needs

Consider interviewing each user using questions like:

- Could you explain in words what you would like a dashboard to convey to you?
- What information do you think is the most important to include?
- When would you want to view this dashboard? All of the time? Start of your day? When someone calls you?
- What type of situation would you like the dashboard to help alert you to? What kind of problem would this represent?
- If you see a problem – is there additional information you would like to see concerning the problem?
- Do you have tools that you use today that you would like to integrate into this dashboard in some way?

For example:

Mike: Is responsible for the Linux servers, and wants to keep a pulse on what he owns

- Needs to be able to view his problem system metrics and events

- Is interested in viewing any of his current service requests and incidents

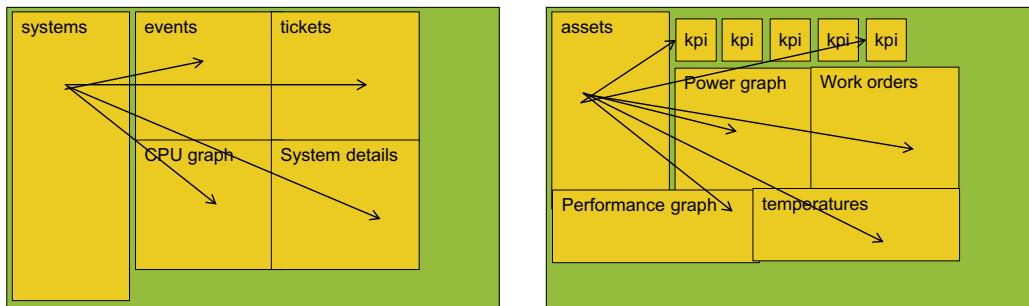
Sarah: Manages the infrastructure team, and wants to visit occasionally to see if there are problems spanning her 3 teams

- Needs to be able to see incidents for her teams
- Wants to see a visual representation of the key business systems that are used by her teams
- Would like access to change management information

Creating prototypes of dashboard pages



- Envision what dashboards you want to start with, focusing on the intended audience
- Create quick storyboards, with simple phrases that explain the intent of each view
- Decide where you want interactivity in the dashboard, and show it with arrows



Creating prototypes of dashboard pages

These charts show information to be presented on a page at a high level example. At this point, you do not need to be concerned with what specific widgets to use but focus on the general layout of the page.

For each view on a dashboard:

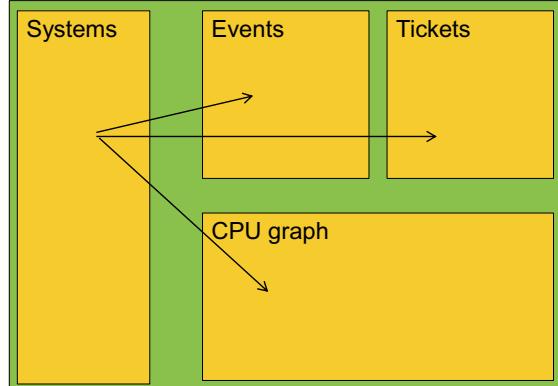
Identify the data source: Where is this data coming from? An IBM product, RDBMS, spreadsheet, web service? What credentials are required to access the data and do they exist?

Refine the dataset: Do you want to see all of the data elements, or a subset? Can you describe the data filter that would be used?

Define navigation links: For views on the receiving end of an arrow, can you describe the query in the context of the item selected on the source side of the arrow.

Example: Operational dashboard

- Systems
 - Source: IBM Tivoli Monitoring 6.3
 - Data set: List, Linux Servers by OS type
- Events
 - Source: Netcool/OMNIbus
 - Data set: List of context events
 - Linking: FQDN from Systems is FQDN from OMNIbus
- CPU graph
 - Source: IBM Tivoli Monitoring 6.3
 - Data set: A graph of CPU history of a selected system
 - Linking: UUID from Systems is UUID in CPU graph



- Tickets
 - Source: Ticket database
 - Data set: List of Linux server tickets
 - Linking: System UUID is Ticket UUID

Example: Operational dashboard

Focus on what is going to be contained within each block. This includes the source of the data, the data set, and how it is linked. Determine if the data will be product provided or where you will obtain the data.

Developing dashboard pages

- Identify and verify data sources
 - Firewalls, networks, and client issues
- Identify authorization roles
 - At the data source
 - In the Jazz for Service Management dashboard server
- Build pages
- Link pages and widgets
- Verify data content and page navigation



Dashboard design fundamentals, DASH overview, and strategy

10

© Copyright IBM Corporation 2016

Developing dashboard pages

Identify the issues at the start of your dashboard project. Obtain an agreement with your user community about what you are going to build and what can be expected.

Evaluating dashboard designs

- Review dashboard data and function with customer
- Adjust content and design as needed
- Repeat process as needed



Evaluating dashboard designs

Ensure that your customer agrees with what you are going to create.

Lesson 2 Visualization services architecture

IBM Training



Lesson 2 Visualization services architecture

Jazz for Service Management components



- Registry services: Centralized data resource locator (Deprecated after version 1.1.2.0)



- **Visualization services (DASH):** Tools to integrate and show data from multiple sources



- **Administrative services:** Tools to automate application management (Deprecated after version 1.1.2.0)



- **Reporting services:** Common reporting infrastructure



- **Security services:** Authentication and single sign-on (SSO) support (Deprecated after version 1.1.2.0)

Jazz for Service Management components

Initially, Jazz for Service Management consisted of these five components. We are covering the visualization services in this course. As of the latest release, Jazz for Service Management consists of the Visualization and Reporting services.

Visualization services overview

The Jazz for Service Management visualization services have the following components:

- A common widget library to create custom dashboards with content from multiple products
- Interactive dashboard pages with widget event connectors
- Customization by systems integrators, administrators, users, and application creators
- User and application single sign-on (SSO) and a unified security model



Visualization services overview

Jazz for Service Management components

The widget library is designed to be applicable to multiple data sources and multiple products. A connector is used to direct data to other widgets on the page. You can build dashboards as you need or prefer.

Data visualization with linked data

- Integrate application data, not applications
- Visualization data is identified and located with HTTP Uniform Resource Identifiers (URIs)
 - Data that is returned from a URI is delivered in a standard, structured data format
 - No application data is transferred through the DASH server
 - Application data is delivered directly to the dashboard widget
- Returned data can include links to other URIs to extend and enrich the available data

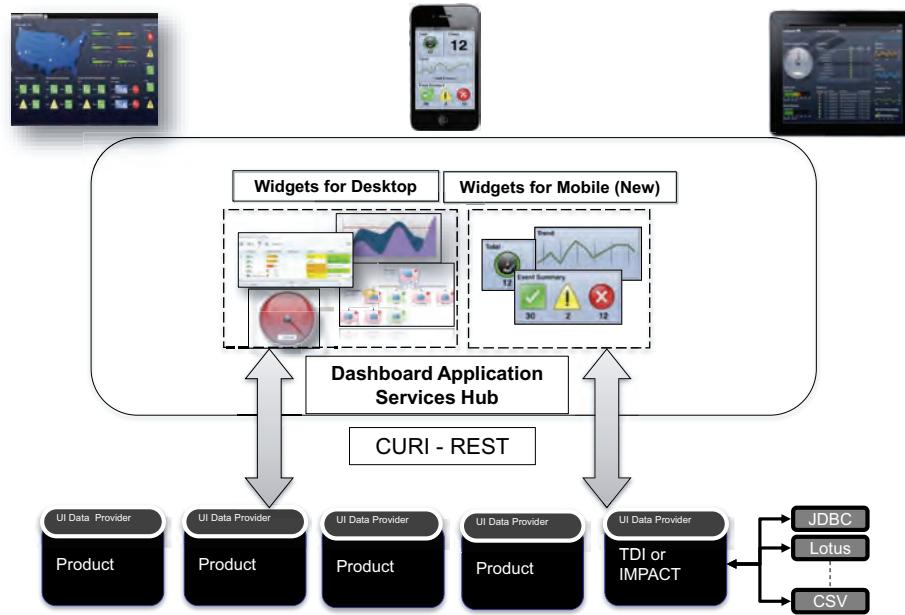


Data visualization with linked data

The latest DASH technology integrates data instead of applications. You can integrate data from many sources. The DASH server manages connections with the data source, and the data source returns a link to where the data can be found. The data never passes through the DASH server and is sent directly to the page. The DASH server acts as a kind of broker.

Some parts of the Tivoli Integrated Portal technology have been brought into DASH.

Visualization services architecture diagram



Dashboard design fundamentals, DASH overview, and strategy

16

© Copyright IBM Corporation 2016

Visualization services architecture diagram

The top graphics show the broad categories of devices that are supported: phones, tablets, computers.

Data providers are like a list of queries. The UI Data Provider standardizes how the widget can see the data.

Jazz for Service Management installation options

- Simple installation:
 - All components are installed on a single server, including DB2
 - Optionally, you can install Tivoli Common Reporting on the same server
- Custom installation:
 - You install one or more components on one or more application servers
- Services are installed with the IBM Rational Installation Manager *
- Installation media includes the Prerequisite Scanner tool to verify that the installation environment supports the services

* Note: Before Jazz for Service Management 1.1.2.0, Tivoli Common Reporting was installed separately with the Tivoli Deployment Engine (DE)

Jazz for Service Management installation options

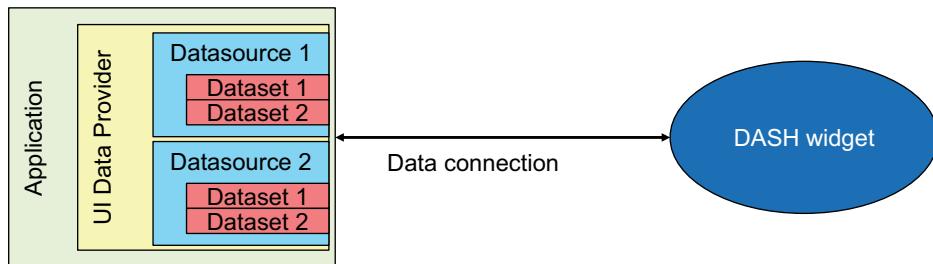
This slide is referring to when Jazz for Service Management contained the initial five components. It technically does not require DB2.

The Rational Installation Manager is a convenient tool to use for the Dashboard software installation.

There are 32-bit and 64-bit libraries that need to be available on the host system, especially for Linux systems.

Providers, data sets, and data sources

- Provider: A software service that provides application or business data to the dashboard
- Data set: A preconfigured set of application or business data; for example, a list of IBM Tivoli Monitoring metrics
 - Data sets consist of one or more rows of data, called items
 - Data sets can include item properties, or attributes
- Data source: A collection of data sets, which are organized at the discretion of the data provider



Dashboard design fundamentals, DASH overview, and strategy

18

© Copyright IBM Corporation 2016

Providers, data sets, and data sources

Provider: Usually a product. This is a software program that is responsible for ensuring data is available through a REST interface.

Visualization: Seen in Quick Edit

Dataset: Group of similar items. These datasets are preconfigured and provided by products. These datasets are like preconfigured queries.

Visualization: table

Item: A data object

Visualization: table row

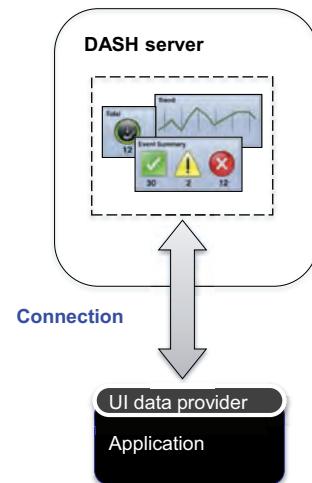
Property: An attribute of the Item

Visualization: table column

The data connection is a specific configuration that you will establish. You create a Connection document per data source or per UI data provider one time. Once you create this connection, it will be available for use continually.

Connections: Widget data pipes

- Connections link the DASH server with an application UI data provider
- Widgets use connections to look up the available UI data provider data sets
- Pointers to data sets are sent as a URL from the UI data provider
- Data is sent directly from the data source to the dashboard widget, not transferred through the DASH server



Connections: Widget data pipes

The widget has a URI that points directly to the data that it needs.

Provider: Usually a product

Visualization: Seen in Quick Edit

Dataset: Group of similar items

Visualization: table

Item: A data object

Visualization: table row

Property: An attribute of the Item

Visualization: table column

Creating connections

- Use the Connections management widget in the Console Settings folder
 - Existing connections are listed in the console workspace
- Click the Create new remote provider icon to create a new connection

The screenshot shows the 'Console Settings' interface. On the left, there's a sidebar with icons for users, catalogs, pages, widgets, views, roles, and help. Below these are sections for 'General' (Catalogs, Connections), 'Console Preference Profiles', 'Pages', 'Widgets', 'Views', 'WebSphere Administrative Console', and 'Export Wizard'. Under 'Roles', there are sections for Group Roles, Roles, and User Roles. A red arrow points from the 'Connections' link in the sidebar to its corresponding entry in the 'General' section of the main menu. Another red circle highlights the 'Connections' link itself. To the right, the 'Connections' section is displayed with a heading 'Connections' and a sub-instruction: 'The connection manager allows you to configure the local and remote connections for this computer.' It includes a note about creating new connections and a table listing existing ones.

Name	Type	Description
Tivoli Directory Integrator	TDI	TDI Generic Data Provider (1.0.21)
tip	tip	Tivoli Integrated Portal Data Provider

Creating connections

From the DASH console, you click the gear icon to open the Console Settings. Select Connections. You see the connections that already exist.

Configuring a connection

- Enter the service provider target connection information
- Click Search to see available service providers
- Select the connection authentication method

The screenshot shows the 'Connections' tab in the IBM Dashboard configuration interface. On the left, there's a 'Server Information' section with fields for Protocol (HTTP), Host name (tbsm152.poc.ibm.com), Port (16310), Path (/ibm/tivoli/test), and a checkbox for 'Connection goes through a firewall' with fields for Firewall address and Firewall port. Below this is a section for credentials with fields for Name (tipadmin), Password, and Confirm password, followed by a 'Search' button. On the right, there's a 'Connection Information' section with fields for Name (TBSM Service Model), Description (TBSM Service Model Data Provider), and Provider ID (TBSM.data.provider). There's also a checkbox for 'Use the credentials of the user (requires SSO Configuration)'. At the bottom are 'OK' and 'Cancel' buttons. A red arrow points from the 'Search' button to a table below. Another red arrow points from the 'OK' button to the same table.

Name	Description	Type	Provider ID
TBSM Service Model	TBSM Service Model Data Provider	TBSM	TBSM
Impact_TBSMCLUSTER		Impact_TBSMCLUSTER	Impact_TBSMCLUSTER

Dashboard design fundamentals, DASH overview, and strategy

21

© Copyright IBM Corporation 2016

Configuring a connection

This is a two step process. You provide enough information to establish the connection. Then you click Search to retrieve a list of possible data providers that matches with the information that you provide. The provider ID must be unique and is often changed from the default ID.

Connection authentication

- Configure service provider authentication in 1 of 2 ways:
 - Use the configured connection credentials for every dashboard user

Use the following credentials to query the remote data providers

* Name:	* Password:
<input type="text"/>	<input type="password"/>
* Confirm password:	
<input type="password"/>	

▪ Use SSO:

- The LTPA authentication cookie for the console user is sent with every data provider request for authentication
- The data provider must be part of the same WebSphere Application Server SSO domain for the LTPA credentials to be accepted

Use the credentials of the user (requires SSO Configuration)

Connection authentication

The name and password is used by default from the Connection document. You can change this to another account, usually another administrator name.

With SSO, the user must have sufficient authority to retrieve the data.

Exercise 1 **Creating dashboard prototype** **diagrams (used in later exercise)**

Exercise 2 **Creating a DASH connection** **document**

Exercises

Complete Unit 1 Exercises 1 and 2 in the exercise guide.

Lesson 3 Creating dashboard pages

IBM Training



Lesson 3 Creating dashboard pages

Page layout styles

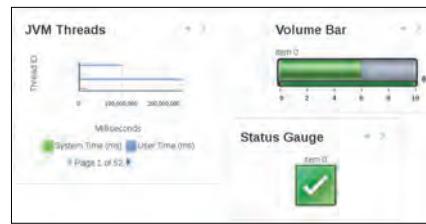
- Before you add widgets to a page, you must select a page layout style
- You choose from three layout styles



Proportional: The widgets can overlap and proportionally change size as the page size changes



Fluid: The widgets are fixed in non-overlapping frames



Freeform: The widgets can overlap but do not change size proportionally

Scalar and list widgets

- Scalar widgets highlight selected column data for one row of a data set
 - Gauges
 - Status gauge
 - Value Status gauge
 - Volume bar
 - Analog gauge
- List widgets show selected column data for multiple rows of a data set
 - List
 - Topology
 - Table
 - Tree Table
- The text widget and refresh widget do not show data from a data set
- Does not apply to web widget content



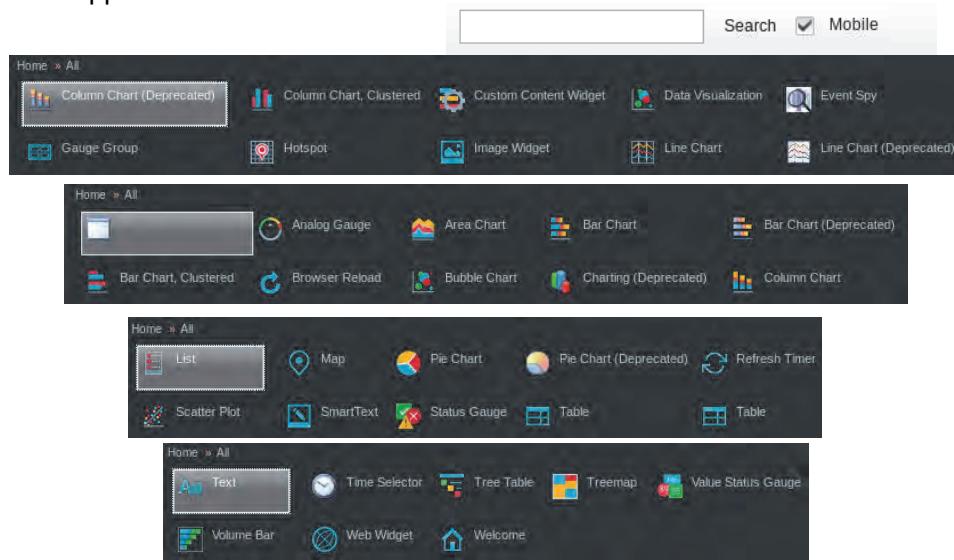
Scalar and list widgets

Scalar widgets are intended to highlight one column of data from one row in a dataset.

List widgets are intended to show one or more rows of data with one or more columns.

Mobile widget overview

Widgets that are supported on mobile devices



Mobile widget overview

With the exception of the topology widget, any of the other widgets can be used for a mobile device. You can use a search field to determine all of the widgets available for a mobile device.

Mobile support

- Web-based approach
 - iOS5 or later
 - Android 4 or later
- Simplified navigation on mobile
- Pages that are created for desktop and mobile can be administered in the same way
- Palette can be filtered for mobile- enabled widgets
- No images on login page
- No toolbar
 - Replaced by a navigation bar, back icons, and page title
 - Single-level navigation slide-out



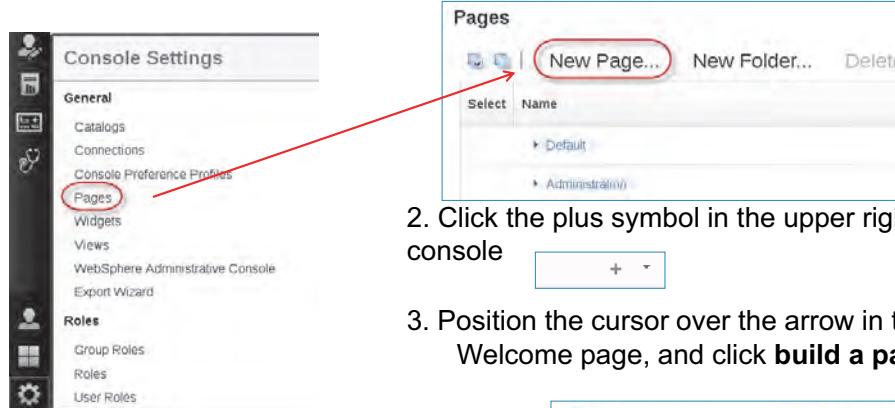
Mobile support

When mobile support is mentioned, a browser connection is made to the DASH server. The display is going to look a little different than when you connect with a desktop. The navigator and buttons will appear differently.

Creating a dashboard page

You have three options for opening the dashboard workspace:

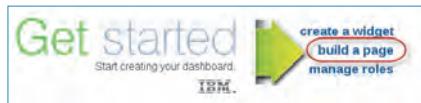
- 1. Click the Pages widget in the Console Settings menu and click New Page



- 2. Click the plus symbol in the upper right of the console

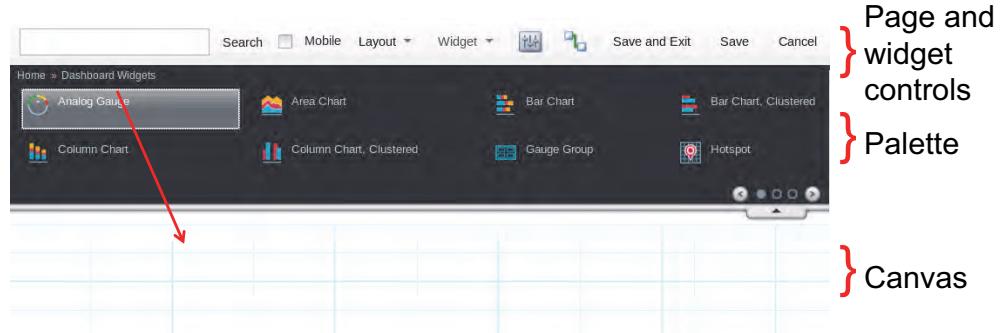


- 3. Position the cursor over the arrow in the console Welcome page, and click **build a page**



There are three ways to create a dashboard page. In the third example, the welcome page can appear differently when other products are installed.

Dashboard workspace elements



- You click a widget in the palette and drag it to the canvas
- You can arrange widgets in freeform or framed configuration, depending on the selected page layout type
- A 20x20 rectangular grid is shown on the canvas to help you arrange widgets

Dashboard workspace elements

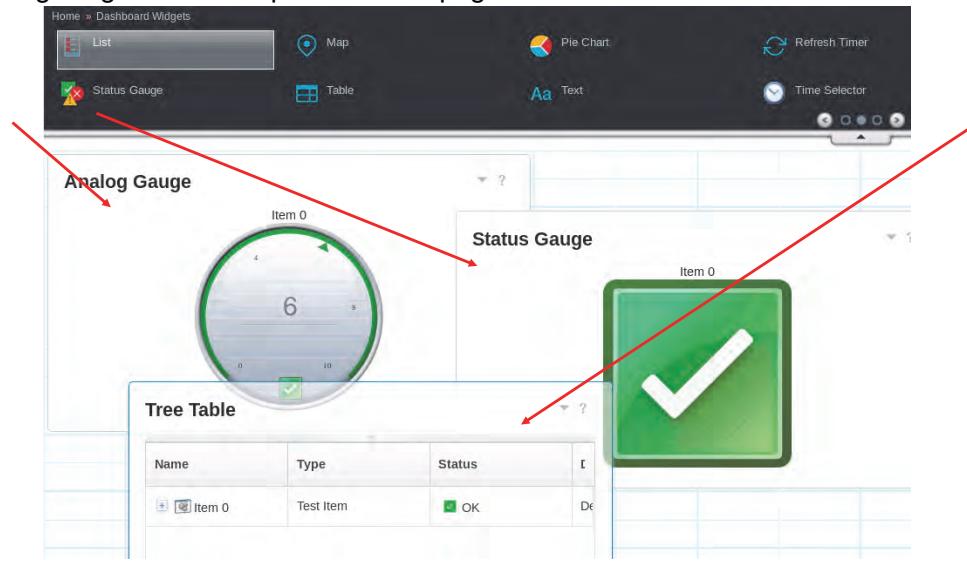
The page editor opens after one of the three methods have been selected.

There are page settings and controls in the tool bar.

If you shrink the size of the browser, that 20 x 20 grid will stay the same.

Adding widgets to a dashboard page

Click and drag widgets from the palette to the page canvas



Dashboard design fundamentals, DASH overview, and strategy

31

© Copyright IBM Corporation 2016

Adding widgets to a dashboard page

Locate a widget, click it, and drag it onto the page. You will see a green icon when you are able to let it go. It will appear red in locations where you can not drop the icon.

In this example, a widget has been added to the page. In the upper right corner of the widget, there are options. Edit is selected.

Next you select a dataset. These datasets are dependent on the connection document that you have made. The search for datasets will communicate with the UI data provider to list datasets.

You select a dataset to go along with the widget, leading to what data you can select for the widget. There can be required selections and optional selections for that widget.

Customizing widgets

- Select a data set for each widget
- Configure how the data set is shown in the widget

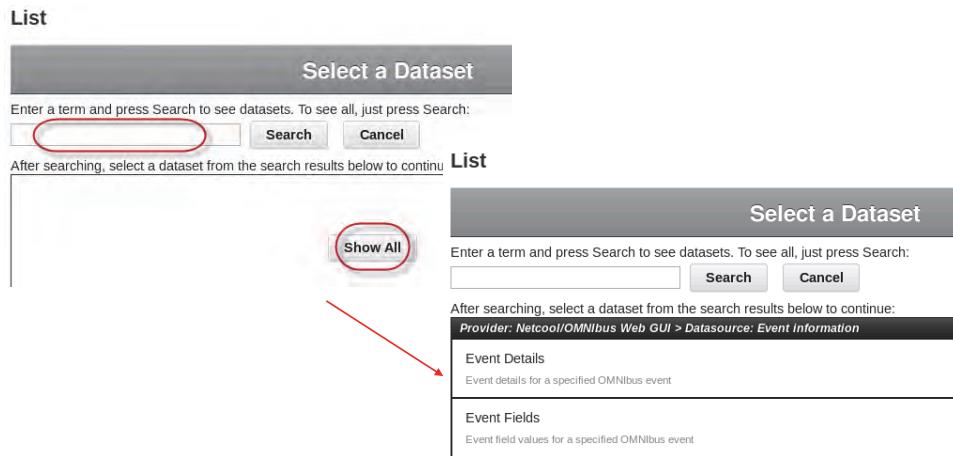
The screenshot shows the configuration panel for an 'Analog Gauge' widget. At the top left is the preview of the gauge with a value of 'Item 0'. To its right is the configuration sidebar with three main sections: 'Required Settings', 'Configure Mandatory Dataset Parameters', and 'Optional Settings'. In the 'Required Settings' section, the 'Value' dropdown is set to 'Event Count'. In the 'Configure Mandatory Dataset Parameters' section, the 'Filter to apply to the ObjectServer' dropdown is set to 'AllEvents (global group)'. The 'Optional Settings' section includes an 'Options:' dropdown with 'Half-moon Gauge' selected. Three red arrows point to specific parts of the interface: arrow 1 points to the 'Edit' button in the top right of the configuration sidebar; arrow 2 points to the 'Event Summary' section in the preview; and arrow 3 points to the 'Select a Dataset' button in the center of the configuration sidebar.

Customizing widgets

An analog gauge widget and an event details dataset is selected.

Selecting a widget data set

- Search for a specific data set or show all available data sets
- Select a data set to continue the widget configuration



Adding widgets to a dashboard page

A list widget and event details dataset is selected.

Customizing widget visualization settings

- Map data set columns to widget attributes
- If they are available, configure more options (optional)
- Visualization settings differ based on the widget, selected data set, and UI data provider

Visualization Settings

Messages: 0

*Required Settings

Map Visualization Attributes to Dataset Columns:

Label You can choose any value	Severity
Status Status type expected	Status
Description You can choose any value	None
Timestamp Date or Date/Time expected. If selected, will be used as default group-by	None

Configure Mandatory Dataset Parameters:

*Filter to apply to the ObjectServer
The filters available to use for this data set

AllEvents (global group)

Visualization Settings

Messages: 0

Optional Settings

Title

Visualization Options:

Group-by Attribute Attribute to sort & group items in List	Timestamp
Sort-by Attribute Initial sort-by Attribute, ignored if Group-by set	None
Sort Order Initial sort order, applicable for grouping also	Descending
Page Size No. of items per page	50

Customizing widget visualization settings

Status and optional settings are provided by you.

Completed dashboard example

- A completed page, that uses the Tivoli Dark console preference profile theme

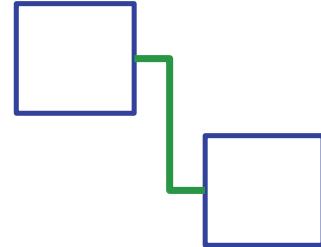


Selecting a widget dataset

This Tivoli Dark console is used a lot due to its striking appearance. However, in a presentation with an overhead projector, it may become harder to see.

Connecting dashboard pages with events and wires

- Most console objects can send state change notifications as internal events
- Events are sent to other widgets on the same page, to another page, or to a widget on another page
- Wires are software communication connections between widgets and pages
- Using events and wires, you configure dashboard navigation and provide visualization context
- For example, clicking a widget opens another page and changes the data that is shown in another widget

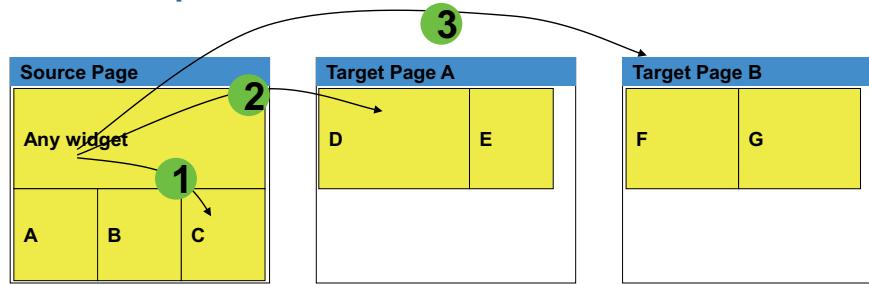


Connecting dashboard pages with events and wires

State change notifications are referred to as events. Functionally, it is like clicking a subroutine call which sends information to another page.

Wires are configured to connect widgets and pages. These are used a lot with pages and widgets.

Events and wires examples



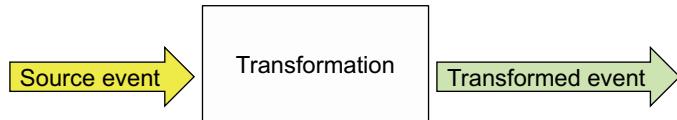
- Example with three wires
 - A NodeClickedOn event is wired to widget C
 - A NodeClickedOn event is wired to widget D, with switchpage=true
 - A NodeClickedOn event is wired to page Target Page B, with switchpage=true
- Result of emitting event
 - Widget C receives NodeClickedOn event
 - Target Page A opens, and widget D receives the NodeClickedOn event
 - Target Page B opens, and widgets F and G receive the NodeClickedOn event

A NodeClickedOn events causes information to be sent to another widget, as in example 1.

The second example shows a NodeClickedOn event so that data is sent to another page.

Event transformations

- Event transformations provide a mechanism to intercept and modify a source event before delivery to a target
 - For example, a source widget can generate only NodeClickedOn events, but a target web widget requires a DisplayURL event
 - An event transformation can be configured to change the NodeClickedOn event to an equivalent DisplayURL event before delivery to the web widget
- Transformations require customization of an XML and JavaScript file

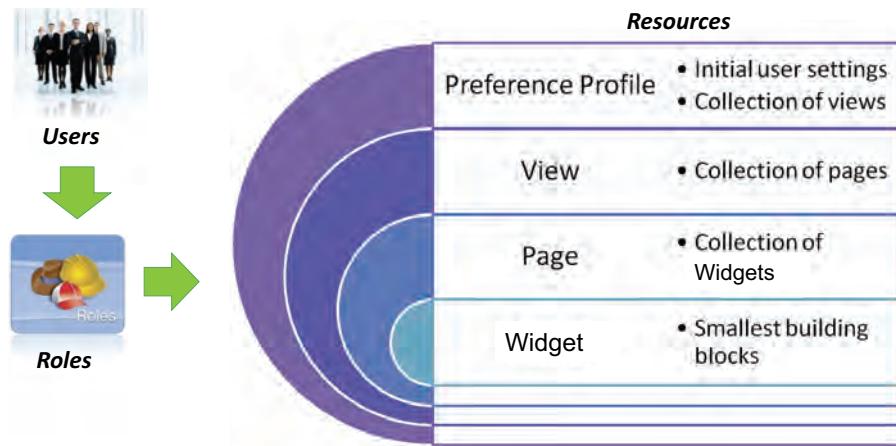


Event transformations

Events between a source and a target can be modified or transformed before the target receives the event. There are easier ways to transform events than using this event transformation method. That will be covered in a later unit.

Managing dashboards

- The interaction of authorization roles that are assigned to users and Jazz for Service Management objects control user access
- When accessing an object, the user roles are checked against the object roles



Events and wires examples

When your dashboards are created, you will want to control who can look at the dashboards or pages. For example, you might restrict users from seeing selected pages. You will probably create pages for specific groups or users.

In preference profiles, roles are assigned to groups or users, determining what the user can see when connecting to the DASH console.

Organizing dashboard pages with views and profiles

- Views: Collections of pages
 - Enable view for mobile devices
 - Control which pages are available for navigation
- Console preference profiles:
 - Control console navigation elements
 - Configure authorization and default view
 - Assign 1 of 3 console themes: IBM OneUI, Tivoli Dark , IBM Design,



Tivoli Dark



IBM OneUI



IBM Design

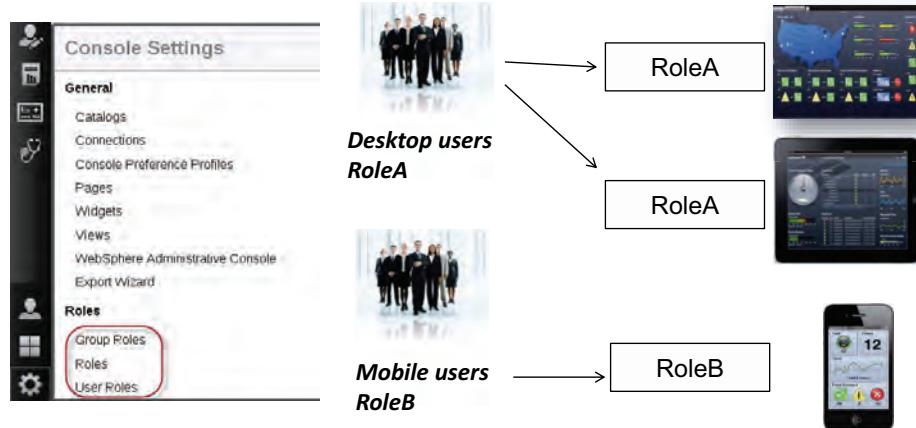
Organizing dashboard pages with views and profiles

In the Views, you can enable a view for a mobile device.

In the console preference profiles, you set the color themes as seen in the three examples on the slide. The IBM Design is more oriented for a mobile device, though it can be used with a desktop.

Authorization roles

- Every widget, page, view, and preference profile must be assigned one or more authorization roles.
- You can create custom roles to simplify page and view management.
- Access levels provide more access control for a role.

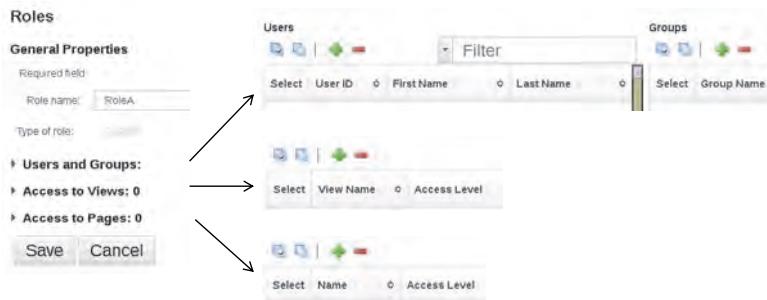


Managing dashboards

From the console settings, you can assign roles to a group or user or define a custom role.

Creating authorization roles

- Click Console Settings > Roles.
- Enter a name for the role.
- Assign the role to users, groups, pages, or views.
- Expand a section and click the plus symbol (+) to select the target object from a list.



Creating authorization roles

Create and assign roles as you need the role.

Managing views

- You use views to define a list of folders and page links.
- You create views with the Console Settings > Views task.
- Mobile devices can use only views with the Enable for Mobile option selected.
- You can add pages and assign view authorization.

The image displays two side-by-side screenshots of a 'Views' configuration interface. Both screenshots show a 'General Properties' section with a 'Required field' checkbox (unchecked), a 'View name' input field (containing 'DesktopUsersView' on the left and 'GtfMobileUsersView' on the right), and a 'Hide any open pages in the work area that are' checkbox (unchecked). The right screenshot has a red circle around the 'Enable for Mobile' checkbox, which is checked. Both screenshots also show 'Type of view' set to 'Custom' and 'View unique name' set to 'CustomViewy3E8g'. At the bottom of each screenshot, there are 'Save' and 'Cancel' buttons. The left screenshot also includes a 'Roles with Access to This View: 2' section and a 'Pages in This View: 1' section.

Dashboard design fundamentals, DASH overview, and strategy

43

© Copyright IBM Corporation 2016

Managing views

This is where you assign one or more pages. Notice on the slide where one is enabled for mobile, where the mobile device can see the pages and views. Without that role in the view to the left, a mobile device could not see the pages and views.

Automatically opening pages in a view

- Expand the Pages in This View section after adding pages.
- Select one or more pages to open (launch) when the view is opened.
 - Each selected page opens in a tabbed window in the console.
 - For views with more than one page, select the top page.

Views

↳ Roles with Access to This View: 2

↳ Pages in This View: 3

Set all pages in this view to launch

Add... Remove Filter

Select	Page Name	Launch	Default
<input type="checkbox"/>	Default	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>
<input type="checkbox"/>	Prop_Test	<input checked="" type="checkbox"/>	<input type="radio"/>
<input type="checkbox"/>	Fluid_Test	<input checked="" type="checkbox"/>	<input type="radio"/>
<input type="checkbox"/>	Free_Test	<input checked="" type="checkbox"/>	<input type="radio"/>

Automatically opening pages in a view

Any pages included in the view can be selected in the launch column and will automatically open. If you have more than one, one page will have to be selected as the default.

Creating console preference profiles

- Use Console Preference Profiles to control navigation elements that a user sees when logged on to the console.
- Create profiles with the Settings > Console Preference Profiles task.
- Assign the console theme.
- Specify one or more views.

Console Preference Profiles

General Properties

= Required field

Preference profile name:

Preference profile unique name:

Theme:

Navigation bar options

Show navigation bar
 Hide navigation bar

Taskbar options

Show taskbar
 Hide taskbar

Console Bidirection Options

Component direction:

Text direction:

Console view options

Show view selector
 Hide view selector

Select the view options the user has access to in the View drop down menu in the banner:

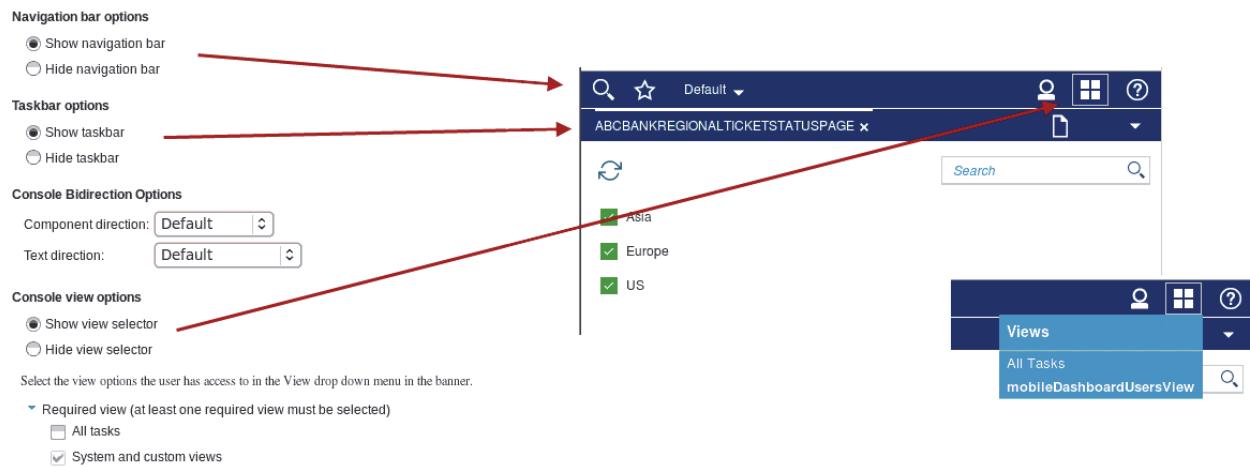
= Required view (at least one required view must be selected)

All tasks
 System and custom views
 Core views
 Favorites

Creating console preference profiles

In the console preference profile, you can set the color theme.

Console preference profile view, task, and navigation bars



Console preference profile view, task, and navigation bars

If you hide the navigation bar, the top blue band will not appear. The show taskbar enables this bar to be seen.

Setting profile authorization and the default view

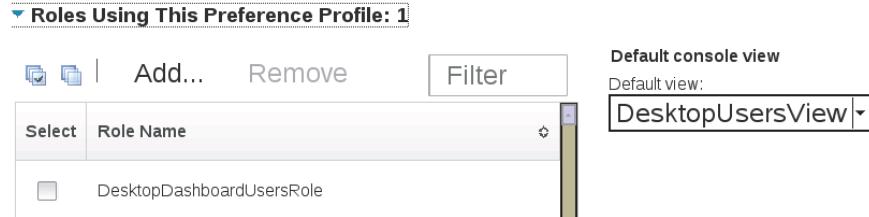
- Specify the default view
- Restrict profile access with authorization roles

▼ Roles Using This Preference Profile: 1

Select	Role Name
<input type="checkbox"/>	DesktopDashboardUsersRole

Filter

Default console view
Default view:
DesktopUsersView



Setting profile authorization and the default view

You specify roles in the profile and then select a default view. That brings up a page automatically. This selection can be easily overlooked.

Exercise 3

Dashboard creation and management basics

Exercises

Complete Unit 1 Exercise 3 in the exercises guide.

Lesson 4 Local web content and dashboard server customizations

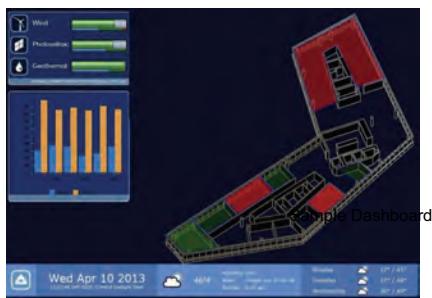
IBM Training

IBM

Lesson 4 Local web content and dashboard server customizations

Hosting local web content

- The DASH server includes a configurable web application to host custom content from the DASH server
- The web application is addressable with the /myBox relative URL path
 - Servlets
 - Images
 - JavaScript, CSS, and other static content



Dashboard design fundamentals, DASH overview, and strategy



- Create URL-addressable UIs in DASH
- Create custom UI with any web-based technology
- View custom UI in the palette like any other widget
- Custom UI on a page can interact with other widgets on the page

50

© Copyright IBM Corporation 2016

Hosting local web content

It is a WebSphere object that gets added in to the DASH server. You use a relative URL, beginning with /myBox. This simplifies the configuration, keeping the information available locally.

Adding files to the myBox web application

- Add files or folders in the /opt/IBM/JazzSM/ui/myBox/web_files/secure or /opt/IBM/JazzSM/ui/myBox/web_files directories
 - Files and folders under the secure directory require console login
- Update the web application with the following command:
 - /opt/IBM/JazzSM/ui/myBox/deployMyBox.sh -v -username <username>
 - -password <password>
 - The command can take up to 5 minutes to complete, but does not require a manual DASH server restart
- Files are addressed with the relative URL /myBox/secure or /myBox
- Example: Use /myBox/secure/us_map.png in web widget



Adding files to the myBox web applications

If you use the secure directory, it requires an SSL authentication with the server. Otherwise, you can access directly from the browser.

The deployMyBox.sh script must be deployed first.

Changing login screen parameters

The screenshot shows the 'Console Settings' interface for the Netcool/OMNibus Web GUI. On the left, there's a sidebar with icons for Dashboards, Catalogs, Connections, Console Preference Profiles, Export Wizard, Dashboard Hub, Pages, Widgets, Views, WebSphere Administrative Console, Console Integrations, and a gear icon for 'Console Properties'. The 'Console Properties' link is highlighted with a red circle. The main panel shows the 'Console Properties' table:

Type	Property Name	Property Value
Editable	LOGIN.COMPANY.IMAGE	ISCPProxy/images/logo
Editable	LOGIN.MAIN.IMAGE	ISCPProxy/images/logo
Editable	ENABLE.CONCURRENT.LOGIN	false
Editable	CONSOLE.NAME	IBM Dashboard Application Services Hub

Red arrows point from the circled property names in the table to the corresponding sections in the 'Console Properties' interface. To the right, a screenshot of the 'IBM Dashboard Application Services Hub' login screen is shown, featuring a logo and fields for User ID and Password.

Changing login screen parameters

Click the gear icon to open the Console Settings folder. Click the Console Properties task. You can modify the DASH server console appearance by changing some property names and the values of the properties.

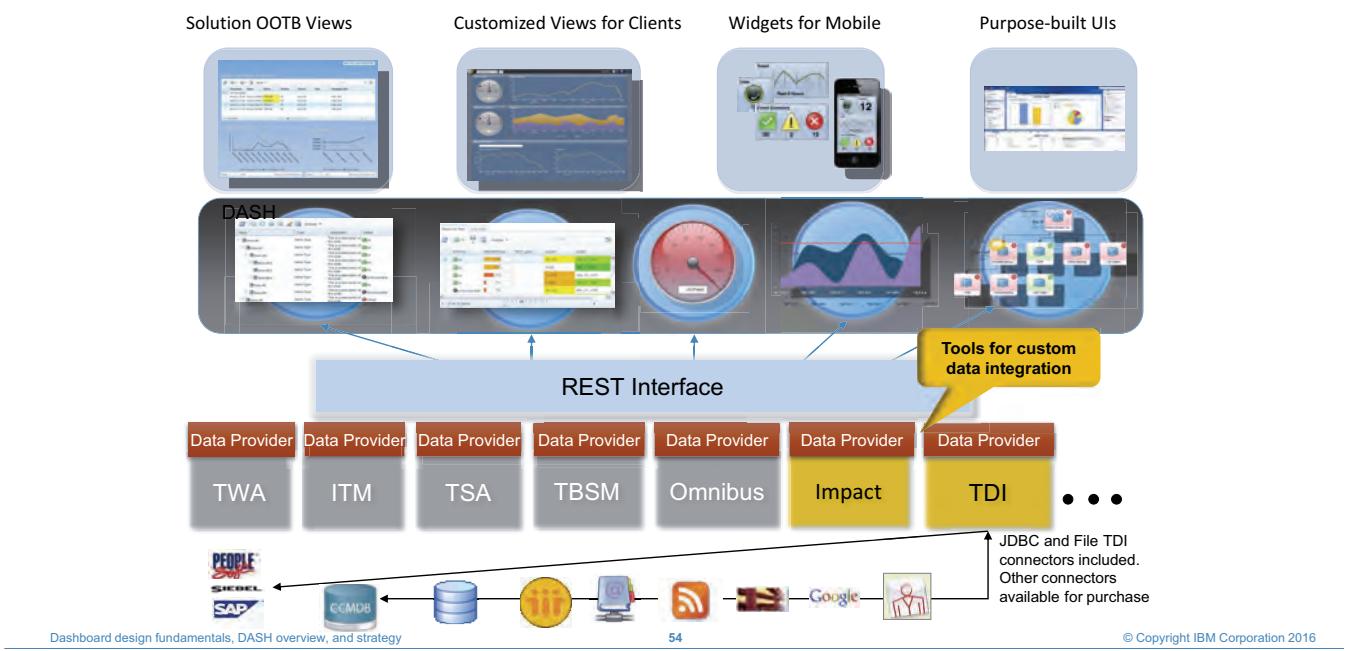
Lesson 5 Visualization services strategy

IBM Training



Lesson 5 Visualization services strategy

Strategy: common data access, common visualizations

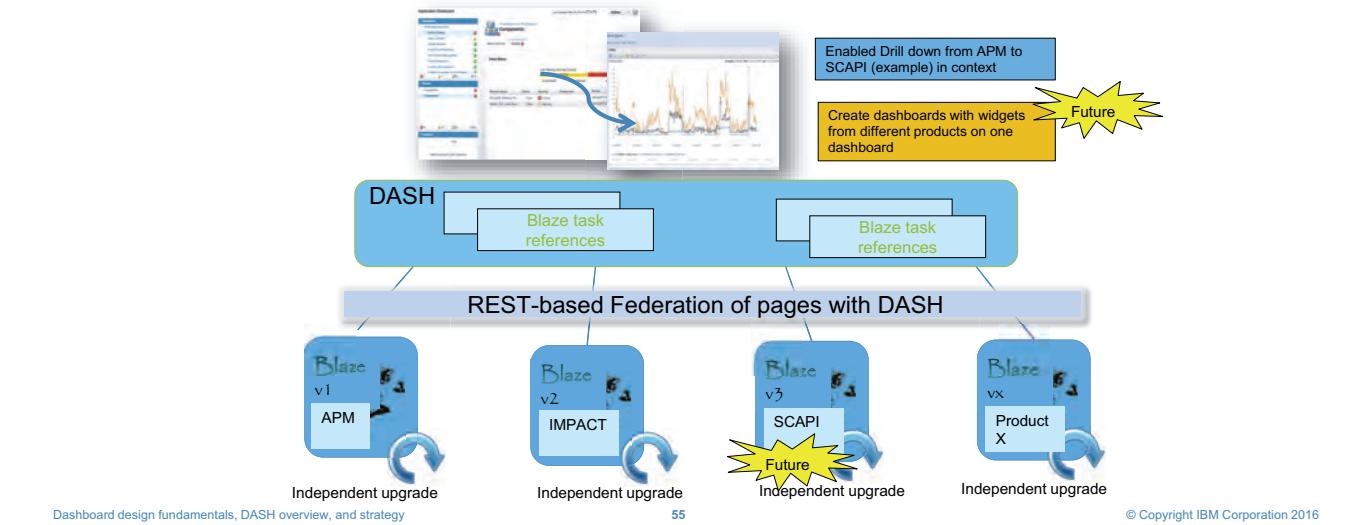


Strategy: common data access, common visualizations

The top level are example dashboards for different types of clients. These widgets communicate through a REST interface through a data provider. You can create your own with Impact and TDI.

Strategy: Blaze technology for stand-alone products

- Lightweight UI services on WebSphere Liberty profile provide core services to visualize and manage application
- Can be used as a stand-alone application UI or federate pages with a DASH server



Strategy: Blaze technology for stand-alone products

Allows independent upgrade/maintenance/release cycles of Blaze based products.

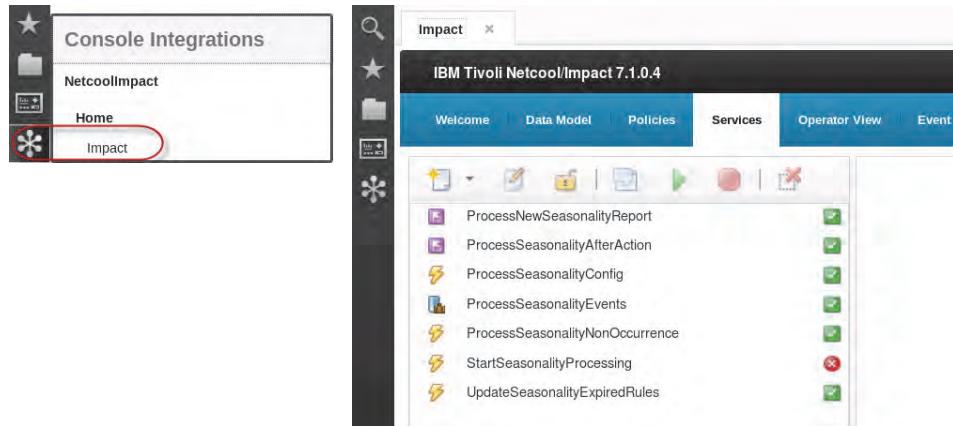
Handshake between Blaze and DASH is a URL.

Blaze technology is a complementary dashboarding technology, based on UI services provided by WebSphere Liberty profile. Consoles are intended to be application specific and provide services just for that application.

The Blaze technology made it simpler for developers to avoid making changes due to changing technologies, as before when using the TIP technology. This looks similar to how TCR integrates with DASH.

Integrating stand-alone console content

- You can integrate supported application consoles into the DASH console view
- Integrated content is added to the DASH console toolbar

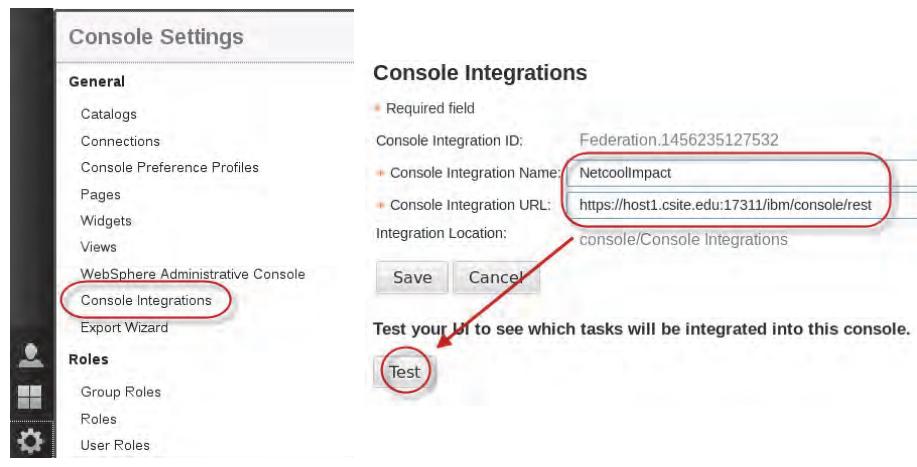


Integrating stand-alone console content

This is an example of Networks for Operations Insight that incorporates with Impact. When Impact is selected, the Impact console opens in the DASH console.

Adding a console integration

- Start the Console Integrations task
- Enter a name for the console and the host and REST interface for the console application



Dashboard design fundamentals, DASH overview, and strategy

57

© Copyright IBM Corporation 2016

Adding a console integration

Go to Console Settings and select Console Integrations. Assign a name and provide a URL. Then click the Test button.

Completing the console integration

- Test the integration and verify that the console tasks are shown
- Save the configuration

Console Integrations
General information regarding the Console Integration being created or edited. Specify the name of your UI, as you would like it to appear

* Required field

Console Integration ID: Federation.1456235127532

* Console Integration Name: NetcoolImpact

* Console Integration URL: https://host1.csite.edu:17311/ibm/console/rest

Integration Location: console/Console Integrations

Test your UI to see which tasks will be integrated into this console.

Status: Connection Successful

The following tasks will be integrated into this console. Pages will be added to the navigation tree under the folder NetcoolImp.

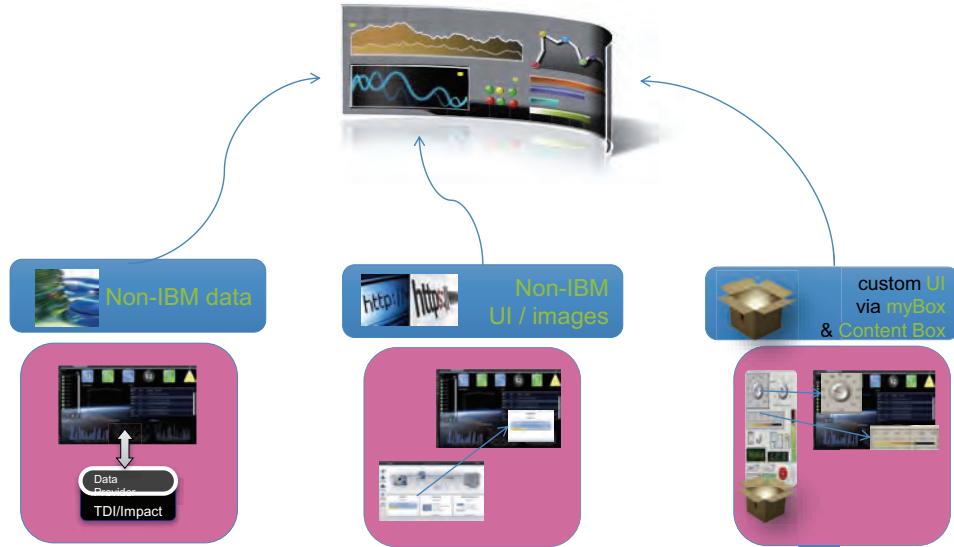
Name	ID	Roles	Supported Platforms	Federated	Type
Impact	impactView	impactAdminUser, impactFullAccessUser, impactOpViewUser, impactMWMAdminUser	DESKTOP	true	page

Completing the console integration

Open Connections to see a list of the pages that have been created in this console.

With the APM UI, you can create pages for APM, which are referred to as applications.

Strategy: Use DASH to integrate IBM and non-IBM application data



Strategy: Use DASH to integrate IBM and non-IBM application data

The overall idea is that if you need to combine data from IBM and non-IBM applications that aren't supported by product provided content, and you want to provide custom content, DASH can bring all of that data together.

Exercise 4 Providing custom dashboard content with ContentBox

Exercise 5 Customizing the console logon page

Exercises

Complete Unit 1 Exercises 4 and 5 in the exercises guide.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Describe the steps that are required to gather dashboard design criteria
- Design high-level dashboard page diagrams based on customer requirements
- Create connection documents to remote UI data providers
- Build interconnected dashboard pages and manage user access to pages, views, and console preference profiles
- Configure locally served web content, such as graphic images, HTML, and JSP pages that can be used in dashboard pages

Unit 2 Tivoli Directory Integrator and DASH basics

IBM Training



Unit 2 Tivoli Directory Integrator and DASH basics

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit reviews the Tivoli Directory Integrator high-level architecture and how it is used to extract and transform data from a variety of back-end data sources. You learn how to use the Tivoli Directory Integrator configuration editor to create AssemblyLines that function as custom UI data providers.

Objectives

When you complete this unit, you can perform the following tasks:

- Describe the Tivoli Directory Integrator components and architecture
- Use the Tivoli Directory Integrator Configuration Editor to create and deploy custom data sets
- Create Tivoli Directory Integrator assembly lines and connectors that read and serve database data and file data
- Configure widgets to show data that is provided with Tivoli Directory Integrator data sets

Lesson 1 Tivoli Directory Integrator overview and architecture

IBM Training

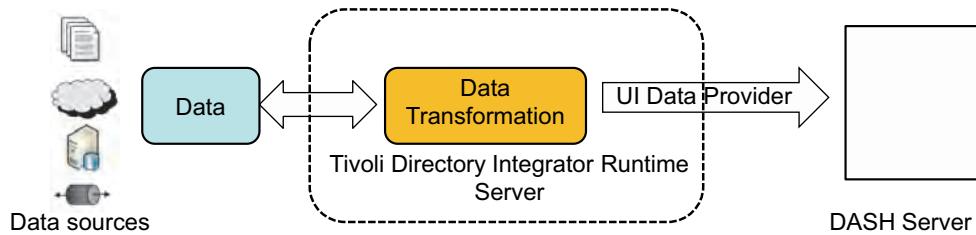


Lesson 1 Tivoli Directory Integrator overview and architecture

3

© Copyright IBM Corporation 2016

Tivoli Directory Integrator overview



- Tivoli Directory Integrator provides tools to integrate, transform, and enhance data from any supported data source
- Data sources include files, directories, databases, collaborative systems, applications, message queues, web services, and REST interfaces.
- Data can be returned to the original data source, passed through more transformations, or delivered to a target system, such as a DASH server.

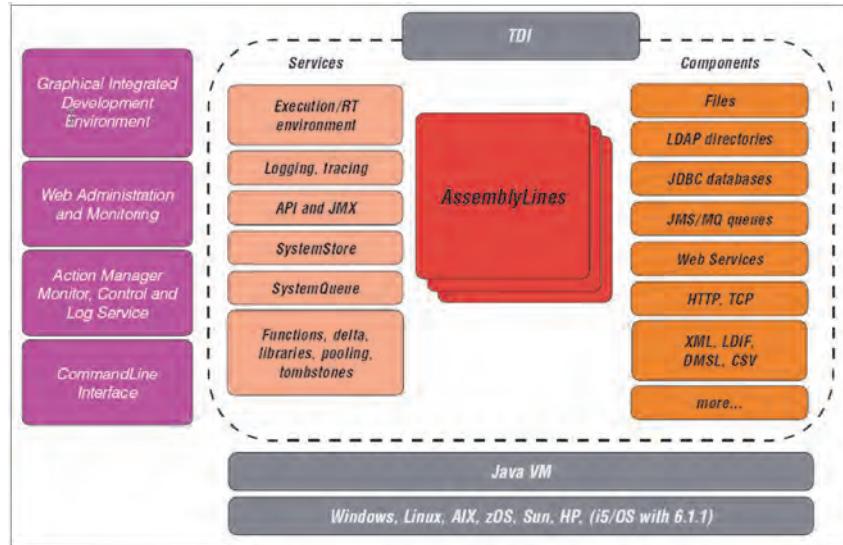
Tivoli Directory Integrator overview

This is a tool that lets you pull data from a data source, transform the data, optionally pass the data to an additional transformation, and then to an output connector. In our environment, the output connector will be the DASH server.

The TDI server is the UI data provider. The data transformations and connections will become your custom datasets.

TDI has connectors that will let you retrieve data from a variety of inputs.

Tivoli Directory Integrator Architecture



Tivoli Directory Integrator Architecture

This is a very high level view of the Tivoli Directory Integrator (TDI) architecture. Inside the DASH component is the run time server. There are tools on the services side. There are components

We will only discuss the TDI box.

TDI has been around a long time and used in many scenarios. It scales well.

Tivoli Directory Integrator and DASH

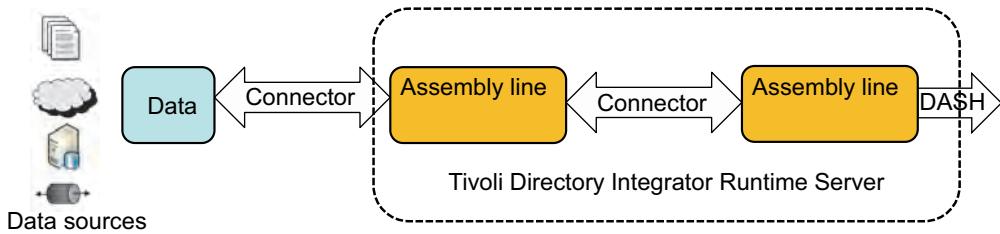
- Used to create custom UI data providers that are not available out-of-the-box.
 - Unlimited license to use with IBM application data
 - Extra license for data from other data sources
- The current release that is included with Jazz for Service Management 1.1.3.0 is IBM Tivoli Directory Integrator version 7.1.1
 - Requires a separate installation and configuration
- The current release is referred to in some documentation as IBM Security Directory Integrator
- This workshop uses the term Tivoli Directory Integrator rather than Security Directory Integrator, though you might see references to SDI when using the software
- Originally created for the security space to manage data updates to LDAP directories

Tivoli Directory Integrator and DASH

Tivoli Directory Integrator is bundled and includes the media, but TDI is not automatically installed.

Tivoli Directory Integrator assembly lines

- A set of software components that are strung together to move and transform data
Assembly lines include data connectors that feed data from a data source to the assembly line components
- Uses JavaScript programming language to control data flow or enhance data
- You can arrange multiple assembly lines in an ordered workflow to apply several data transforms
- An assembly line can be triggered when a connected data source changes



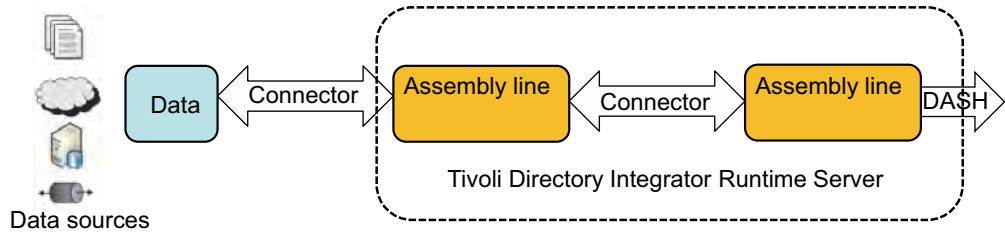
Tivoli Directory Integrator assembly lines

The heart of does is called an assembly line. You can chain multiple assembly lines together to enhance data. Ultimately, the data is handed to an output connector which is the DASH server.

It does not require expert knowledge of Java scripting, though it helps to understand scripting languages in general.

Data connectors

- Data enters the assembly line from a data source with a component called a Connector
 - Record-oriented data is mapped to attributes with the data source schema
 - Non-record data streams are mapped to attributes after transformation with a prefix component called a parser
- Connectors are configured to read input from, or send output to, a data source



Start by configuring a data connector that provides input to the assembly line.

Connector modes

Connector input is configured in one of several modes:

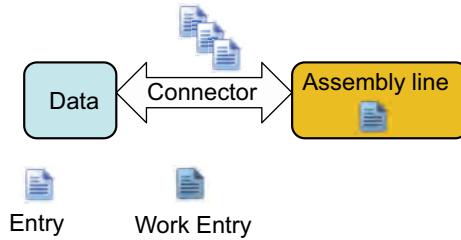
- AddOnly: This mode adds data entries to a data source.
- Iterator: Used to iterate through a data source one record at a time.
- Delete: Used to delete records from a data source.
- Lookup: Used to join data from different data sources.
- Update: Used to add and modify records in a data source.
- Delta: Used to apply incremental changes to a data source, based on delta operation codes.
- CallReplyMode: Used to make requests to data source services that require input parameters and to receive reply return values.
- Server: Used to wait for an incoming event, dispatch a responding thread, and send a reply to the originating source.

Connector modes

Connectors support any of the modes listed on the slide. This course uses the Iterator mode frequently. Records are processed from start to finish with each being processed by the assembly line. The other modes provide versatility.

Connector data: Entries and attributes

- Connectors convert data into a consistent form that is processed by an assembly line
 - Tivoli Directory Integrator connectors are different from DASH connectors
- As a record is read from a data source, the record is referred to an entry, similar to a variable name
- The value or values that are assigned to the entry (the record data) are referred to as attributes
- Any entry and attributes that are processed by an assembly line are referred to as the Work Entry or Work Object



Connector data: Entries and attributes

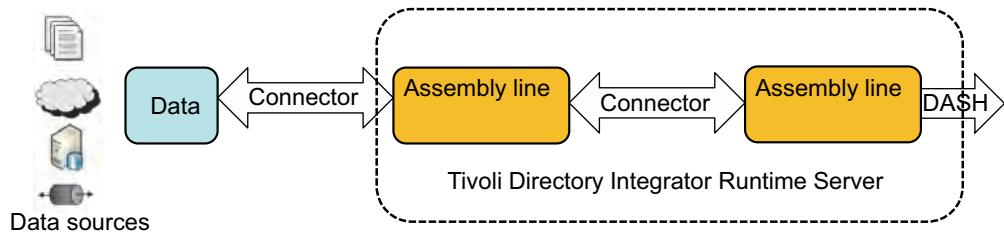
DASH widgets are generally passive objects. You pass data to the widget and it displays the data.

Terminology:

- As records are read into a data connector, the record is referred to as an entry.
- The individual values assigned to an entry are called attributes.
- Generally speaking, the output of the Work Object will be passed as the dataset that you are creating.

Assembly line data flow

- Data in an assembly line can be modified, enhanced with a supplemental data source, or passed to another assembly line or output connector.
 - The data connector passes data to the assembly line one record at a time
 - The data that is provided to an assembly line by a connector is referred to as the work object.
 - Data is accessed in an assembly line as JavaScript variables
- The final work object is passed to an output connector
 - When used with DASH, the output connector is the UI data provider to the DASH server



Assembly line data flow

A mode is selected, like iterator.

Technically, what is being passed are links or pointers to the data, not the data itself.

Building assembly lines

- Assembly lines are developed with an Eclipse-based tool called the Configuration Editor (CE)
- All components in an assembly line are automatically registered as script variables
 - Example: A DB2 connector would register a database column called EMPLOYEES as the variable name conn.EMPLOYEES
- Use as few assembly lines as possible
 - For example, use branches or switches to define multiple operations handled by a single assembly line

Building assembly lines

The tool for TDI is a complete development environment.

conn.EMPLOYEES is a connector for that database column.

A single assembly line is more efficient.

Runtime server

- Deploy completed projects and assembly lines by adding them to the runtime server component
- You can start assembly lines from the command line:
 - `ibmdisrv -c <name of configuration file> -r <Assembly line>`
 - For example, use the runtime server in scripts or crontab
- Alternatively, drag the project to the server icon in the configuration editor



Tivoli Directory Integrator and DASH basics

13

© Copyright IBM Corporation 2016

Runtime server

You have access to the runtime server where the data connectors and assembly lines happen.

In production, you would use an rc.d or crontab script to start the runtime server or Assembly Line.

Tivoli Directory Integrator - DASH sites for reference

IBM TDI Getting Started:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDI.doc_7.1.1/gettingstarted.pdf

YouTube channel: http://www.youtube.com/view_play_list?p=DA11D0AF46C951D7

TDI User Group: <http://tdi-users.org/>

Google Public "best effort" support forum:

<https://groups.google.com/forum/?fromgroups#!forum/ibm.software.network.directory-integrator>

Shortcut : tinyurl.com/dashtdi

If you want more details about TDI, these links can help.

Lesson 2 Using the Configuration Editor

IBM Training

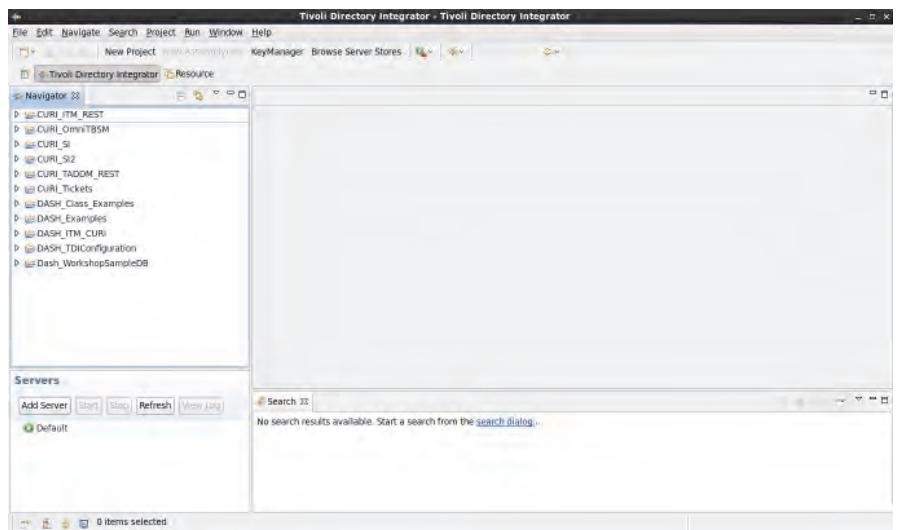


Lesson 2 Using the Configuration Editor

Starting the configuration editor

Start the configuration editor with the following command:

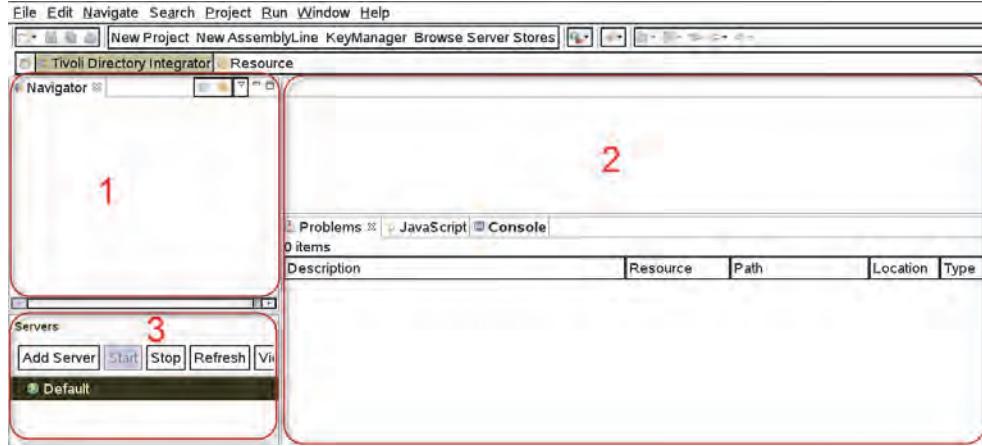
- \$TDI_Install_Dir/ibmditk
- Example:
/opt/IBM/TDI/V7.1.1/ibmditk



The configuration editor is Eclipse based.

Configuration editor interface

- Navigator section: A tree list of projects and project components
- Editor workspace section: Context-driven view of project component details and configuration
- Tivoli Directory Integrator Runtime Server management section: Use to start, stop, and pause project operation



Tivoli Directory Integrator and DASH basics

17

© Copyright IBM Corporation 2016

Configuration editor interface

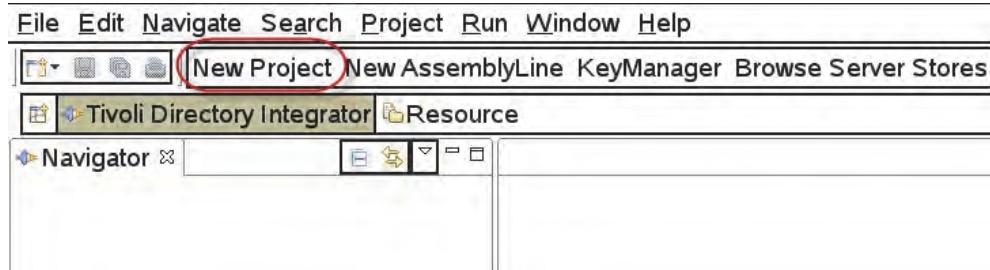
Project components can include assembly lines, scripts, and more choices.

If you click a component in the navigator, it opens the area in Section 2. The third section is used to manage the dataset that is created.

Creating a project

Click New Project

- All required project component structures are automatically created and listed in the Navigator section



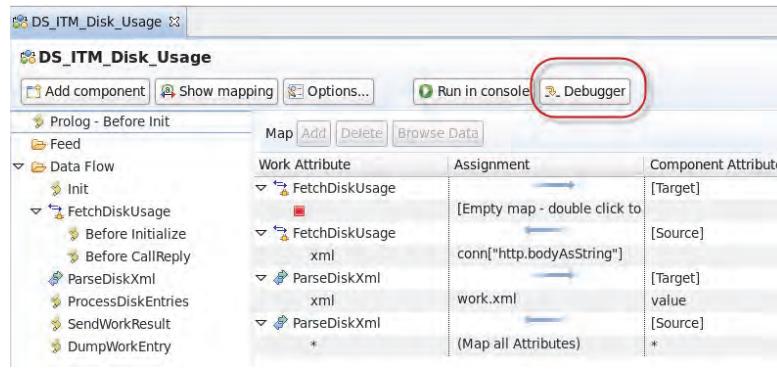
Creating a project

Start by creating a new project.

Configuration Editor debugger

The configuration editor includes a data flow debugger

- Use to check values at various points in an assembly line operation during development
- You can debug scripts line by line
- You can configure conditional breakpoints
- You can watch and manipulate the value of variables



Tivoli Directory Integrator and DASH basics

19

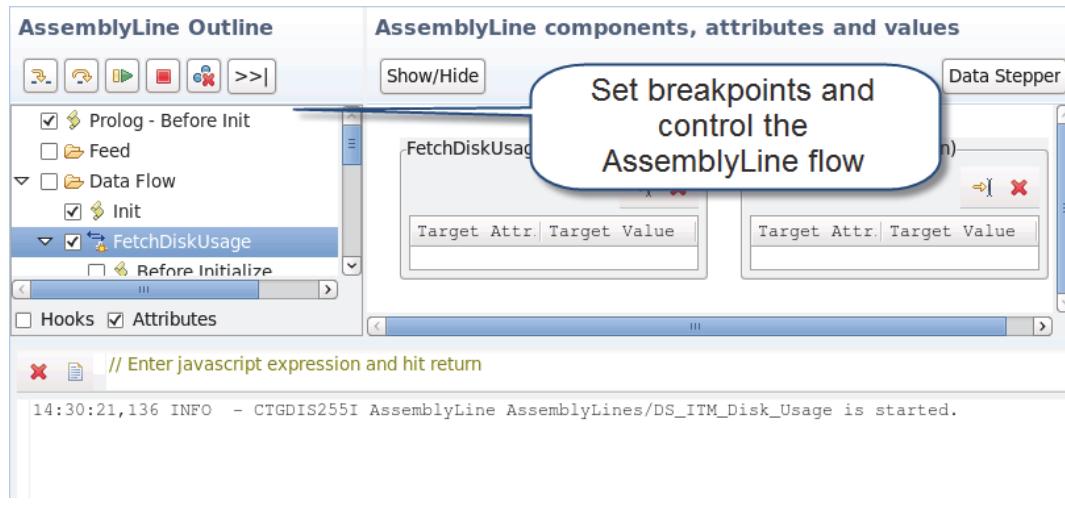
© Copyright IBM Corporation 2016

Configuration Editor debugger

This is a fully functional development environment. You can set check points. You can step through scripts one line at a time.

Breakpoints

- You can set configure breakpoints at every point of the assembly line operation
- Breakpoints can be conditional



Tivoli Directory Integrator and DASH basics

20

© Copyright IBM Corporation 2016

Breakpoints

Break points can help you debug your assembly lines.

Debugger watch list

- Use the watch list to view entry attributes as the assembly line is processed
- You can expand data structures and modify and update attribute values

Watch List	
Edit Watch List	
Name/Expression	Value
work	
conn	
▷ Global variables	
▷ Watch List	

Lesson 3 Creating an AssemblyLine

IBM Training



Lesson 3 Creating an assembly line

22

© Copyright IBM Corporation 2016

Naming DASH Tivoli Directory Integrator projects

- Click New Project at the top of the configuration editor and assign a project name
- DASH TDI project names must begin with the prefix DASH_
 - The project name is seen in the DASH server as the UI data provider Data Source



Naming DASH Tivoli Directory Integrator projects

Start by creating a new project. The name must begin with DASH_.

When you select a dataset, it contacts the TDI server, requests a list of all projects and assembly lines. If it doesn't begin with DASH_, it won't be recognized as a valid datasource.

DASH assembly line naming standards

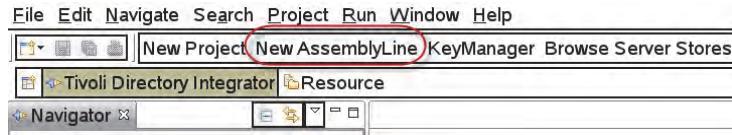
Pattern	Description
DASH_*	Used in project name to identify a CURI data source to DASH
DS_*	Identifies the assembly line output to DASH as a CURI data set
DS_*_columns	Helper assembly line; provides column details for DS_*
DS_*_itemstyles	Helper assembly line; provides item styles for DS_*
DS_*_linkstyles	Helper assembly line; provides link styles for DS_*
DS_*_relateditems	Helper assembly line; provides related items for an item in DS_*
DS_*_tasks	Helper assembly line; provides tasks for items in the DS_*
DS_*_events	Helper assembly line; provides change event data for a DS_*
DS_*_parameters	Helper assembly line; provides parameters for a DS_*
DS_*_relationships	Helper assembly line; provides relationship data for an item in a DS_*
DS_*_links	Helper assembly line; provides link data for an item in a DS_*
DS_*_status	Helper assembly line; provides a list of status items
DS_*_paged	Helper assembly line; identifies the DS_* as a large table mode (paged) data set

DASH AssemblyLine naming standards

The exercises will use some of these naming standards.

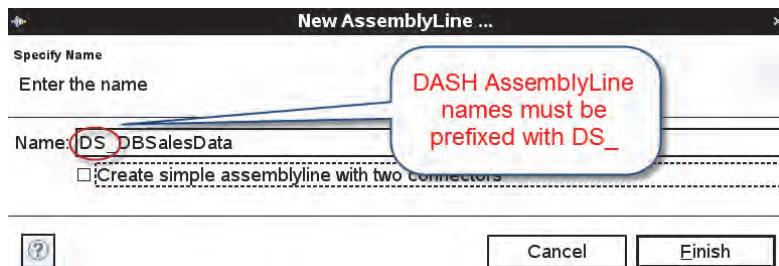
Adding an assembly line

- Click New AssemblyLine



- Prefix the name with DS_

- The assembly line name is the data set name that is used when configuring a widget



Tivoli Directory Integrator and DASH basics

25

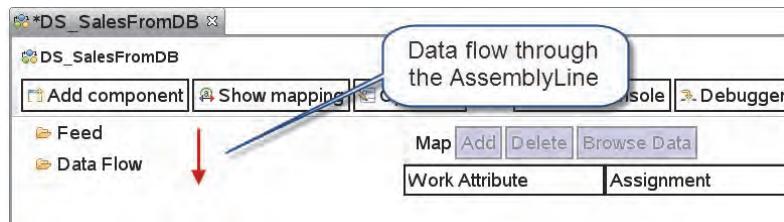
© Copyright IBM Corporation 2016

Adding an assembly line

While the project is selected from the navigator tree, click New Assembly Line. It must have DS_ as the prefix. The name following the DS_ will be the dataset name that you search.

Assembly line data flow in the Configuration Editor

- The assembly line workspace shows a Feed and Data Flow object
- Generally, the feed is configured to pull data from a data source and the Data Flow defines how the source data is modified
 - Think of data as moving from the Feed through the Data Flow
- The data is mapped to one or more Work Attributes, which are delivered as assembly line output attributes
 - Output attributes can be configured as input attributes for more assembly lines
 - The DASH server is usually the final output attribute destination



Assembly line data flow in the Configuration Editor

You see in the workspace the folders Feed and Data Flow.

For most assembly lines, the flow will be from the feed to the data flow sections.

As data is mapped from a Connector, you assign work attributes.

Lesson 4 Creating a UI data provider data set from a database

IBM Training

IBM

Lesson 4 Creating a Tivoli Directory Integrator-based data set from a database

27

© Copyright IBM Corporation 2016

Example: Show sales database data in a chart widget

- Car Sales Company
 - Use this fictitious business to drive different use cases
- The database contains sales data for three different kinds of vehicles:
 - Sedans (Hybrids and Sport)
 - Vans (8 and 10 passenger)
 - Pickups (1/2-ton and 1-ton)
- Sales figures are recorded summarized from two different locations
 - California
 - North Carolina



Example: Show sales database data in a chart widget

This Car Sales example is used several times in the course presentations and exercises.

Database data set creation tasks

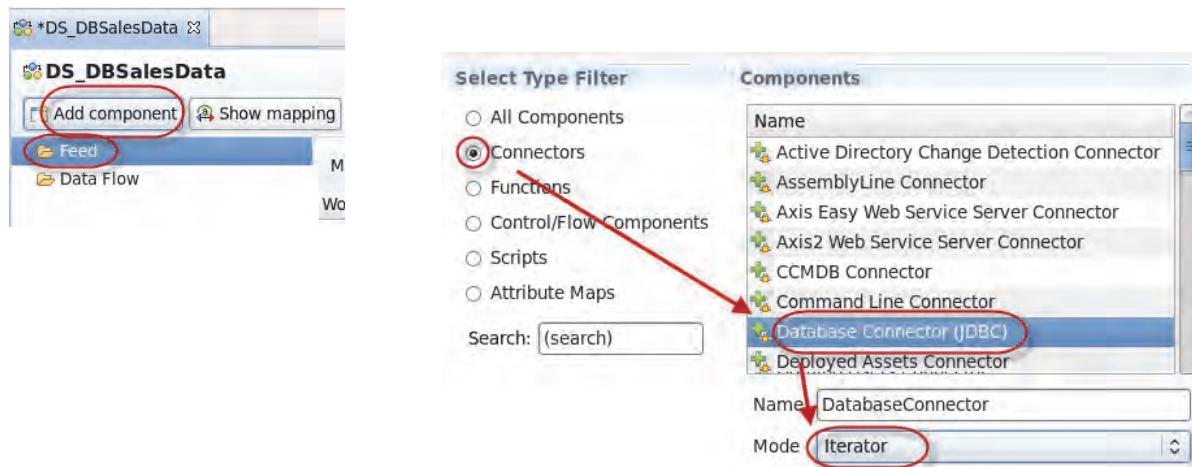
- Complete the following tasks to create a data set from a database:
 - Create and configure a data connector to the source database
 - Map the connector work entries to the data source columns
 - Deploy the assembly line to the runtime server
 - Test the UI data provider with a dashboard widget

UI data provider creation tasks

We create a dataset, also called a UI data provider, from the database.

Creating a DB2 database connector

- Click Feed in the workspace and click Add component
- Select the Connectors filter and select Database Connector (JDBC)
 - Set the Mode to Iterator



Tivoli Directory Integrator and DASH basics

30

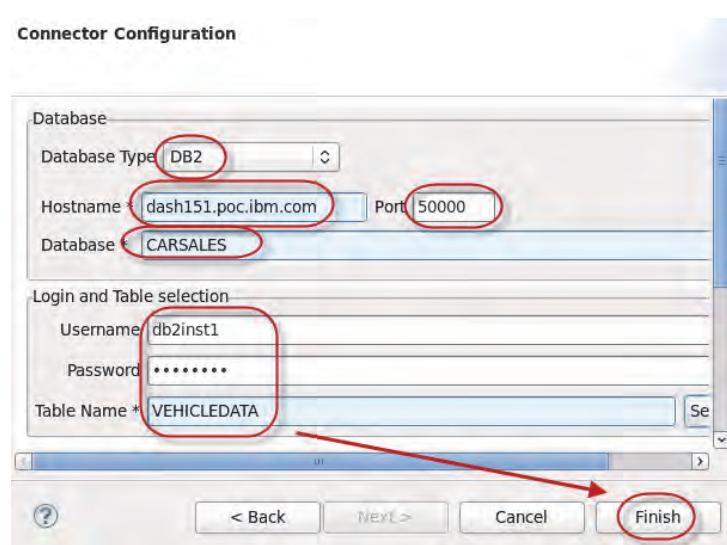
© Copyright IBM Corporation 2016

Creating a DB2 database connector

Clicking Feed and Add Component provide a list of all of the component types.

Configuring a DB2 database connector

- Select DB2 as the Database Type
- Enter host name
- Enter port
- Enter database name
- Enter authorized user name
- Enter password
- Enter source table name
- Click Finish



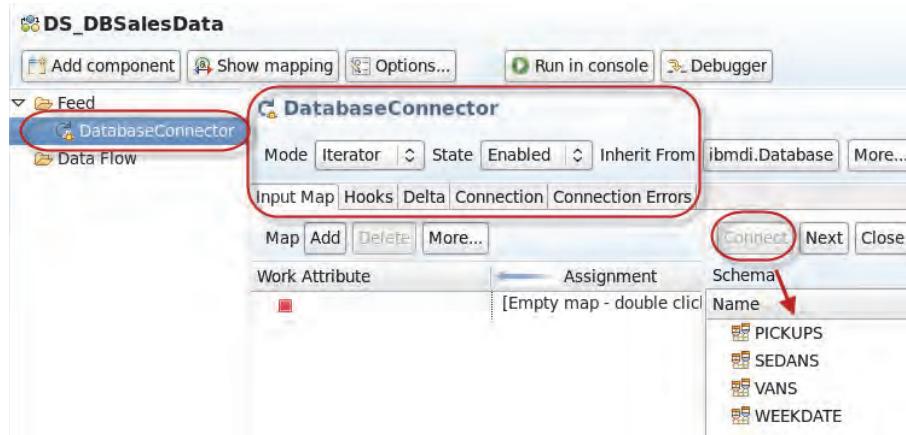
Configuring a DB2 database connector

Configure how to get to the database by providing the information required by the Connector Configuration.

There is a Select button to the right of the Table Name. Using Select will retrieve the data to verify that the configuration is correct.

Verifying the database connection

Click Connect to connect to the configured data source



Verifying the database connection

The Database Connector is displayed. By clicking the Connect button, it will make the database connection. Then clicking Next will iterate through the records.

The columns in the table need to be assigned to the work attributes.

Previewing data

After connecting to the data source, you can iterate through the data source records:

- Click Next to iterate data from the data source
- As data is read, the schema is automatically identified and the input streams are assigned to be processed through a corresponding Java class interpreter

Name	Sample Value	Required	java Class	Native Syntax
PICKUPS	8	Optional	java.lang.Integer	INTEGER
SEDANS	10	Optional	java.lang.Integer	INTEGER
VANS	5	Optional	java.lang.Integer	INTEGER
WEEKDATE	2012-10-05	Optional	java.sql.Date	DATE

Previewing data

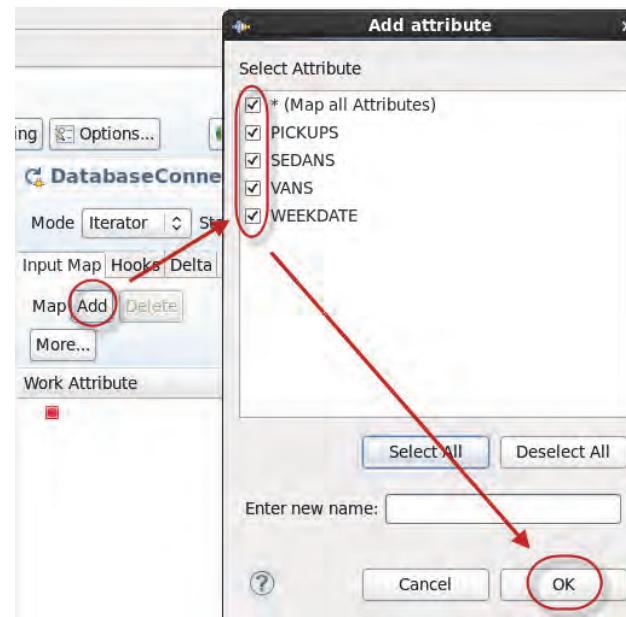
When stepping through the records, it picks up the schema information and data types automatically.

Some widgets have data type requirements, like numeric or character data, making the data type important to assign correctly for the widget.

Mapping data source columns to assembly line output

Map data connector input values to output work attributes

- Click Add in the Input Map tab
- You can map one or more input attributes



Mapping data source columns to AssemblyLine output

This begins to show how to map the connection to the work object.

Click Add to see the available attributes. Select All , or select individual attributes by clicking boxes.

If you wanted to join from different databases or tables, then you create a second component. You map in the components, and from the javascript area, you can join the data

Reviewing output mappings

- Click Show mappings to see a summary of the attribute mappings
- Finally, the mapping view shows the general flow of data from the data source, through the connector, and to the Work Attribute mappings

The screenshot shows the 'DS_DBSalesData' configuration window. At the top, there's a toolbar with 'Add component', 'Show mapping' (which is circled in red), 'Options...', 'Run in console', and 'Debugger'. Below the toolbar, there's a navigation bar with 'Feed' and 'DatabaseConnector' under 'Data Flow'. A 'Map' button is also present. The main area displays a table titled 'Work Attribute' with columns 'Assignment' and 'Component Attribute'. The table lists several entries, with the first entry '*(Map all Attributes)' expanded to show detailed assignments for attributes like PICKUPS, SEDANS, VANS, and WEEKDATE.

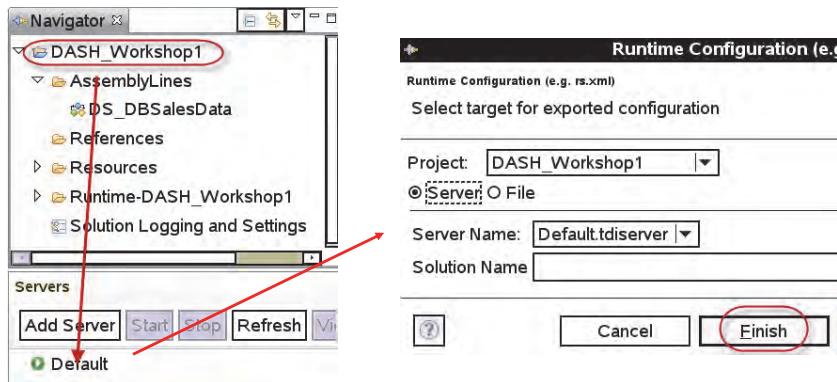
Work Attribute	Assignment	Component Attribute
*	(Map all Attributes)	[Source]
PICKUPS	conn.PICKUPS	*
SEDANS	conn.SEDANS	PICKUPS
VANS	conn.VANS	SEDANS
WEEKDATE	conn.WEEKDATE	VANS

Reviewing output mappings

Clicking Show mappings shows the work attributes and assignments.

Deploying the assembly line to the runtime server

- Save the project, then drag the project name in the navigator to the Default server icon in the Servers section
 - The mouse cursor changes shape when the drag operation is enabled
- Click Finish to confirm the project export to the runtime server



Deploying the AssemblyLine to the runtime server

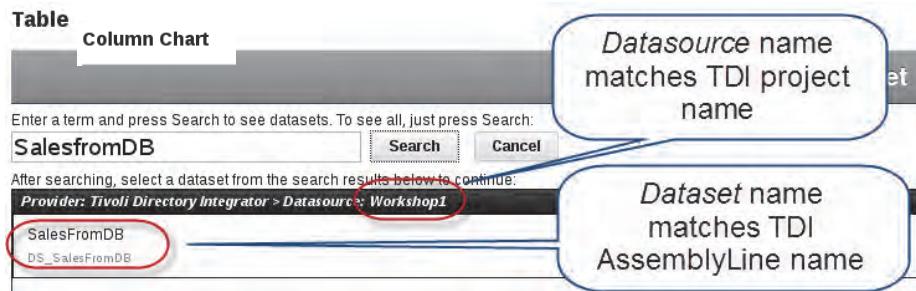
The Assembly Line is saved.

To deploy the assembly line to the runtime configuration, you drag it. On some of our hosting sites, it might be a little difficult to grab the project name, and can require you to pause for a few seconds.

Using the data set with a dashboard widget

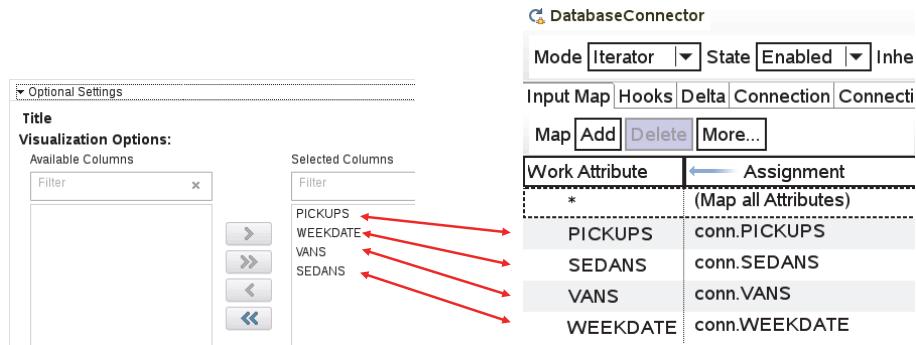
In this example, editing a chart widget, search for data sets that match SalesfromDB.

- Select the SalesFromDB data set
- The DS_SalesfromDB assembly line output data is available for delivery to the table widget



Correspondence of data set values and work attributes

The Work Attribute values in the assembly line correspond to the column values in the data set.



DASH widget configuration

Tivoli Directory Integrator assembly line input map

Correspondence of data set values and work attributes

The slide shows the data mappings from the DASH side on the left and the TDI side on the right.

This example is using a table widget. When you see available columns, the database connector area is where it is coming from.

Configuring the column chart

For this example, map the WEEKDATE values to the X axis and stack the PICKUPS, VANS, and SEDAN values to a column on the Y axis.

* Required Settings

X axis field
The data column to be plotted as the independent variable.

Y axis value field
The data column to be plotted as the dependent variable.

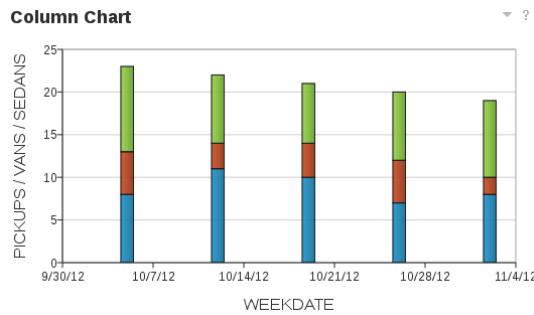
Group by field
The data column used to collect data into sub-sets when stacking multiple sets on a chart. Ignored when plotting multiple columns from each row.

Configuring the column chart

The three Y-axis choices are made to produce a stacking bar chart.

Completed DASH widget

The completed chart widget, as seen in the dashboard page.



Completed DASH widget

Though it is not displayed on this slide, you can merge information from multiple data sources into a single provider.

Instructor demonstration

- Creating a UI data provider data set from a database



Exercise 1

Creating a UI data provider data set from a database

Student exercises

Complete Unit 2 Exercise 1 in the exercise guide.

Lesson 5 Creating a Tivoli Directory Integrator-based data set from a CSV file

IBM Training



Lesson 5 Creating Tivoli Directory Integrator-based data set from a CSV file

Example: Creating data sets from unstructured data

```
WEEKDATE,VANS,SEDANS,PICKUPS
2012-10-05, 5, 10, 8
2012-10-12, 3, 8, 11
2012-10-19, 4, 7, 10
2012-10-26, 5, 8, 7
2012-11-02, 2, 9, 8
```

- In this example, one of the car sales offices maintains the sales data in a CSV file
- You must create an assembly line that reads the data from the CSV file and then use the resulting data set to show the data in a column chart on a dashboard page

Example: Creating data sets from unstructured data

This example shows you the same numbers that you used in the previous exercise when you accessed the database. This time, the numbers are from a CSV file.

CSV file-based data set creation tasks

Complete the following tasks to create a CSV UI data provider:

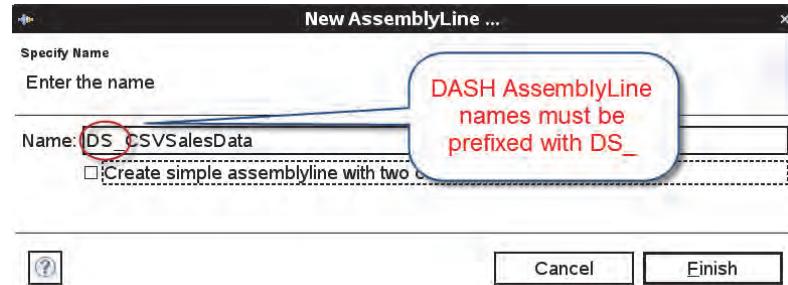
- Create and configure a file data connector to the source file
- Map the work entries from the connector to the appropriate file attributes
- Modify the work entries to match the appropriate data types (date and integer)
- Deploy the assembly line to the runtime server
- Test the UI data provider with a dashboard widget

CSV file-based data set creation tasks

Modifying the work entries is required sometimes because of the differences in character or numeric handling.

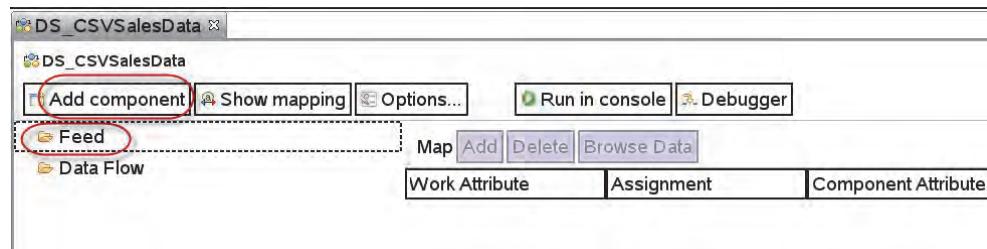
Adding an assembly line to an existing project

- Tivoli Directory Integrator projects can include multiple assembly lines
- In this example, create an assembly line to read data from a CSV file
 - Click **DASH_Workshop1** in the Navigator and click **NewAssemblyLine**
 - Enter **DS_CVSSalesData** in the assembly line name field and click **Finish**



Adding a file connector to the assembly line

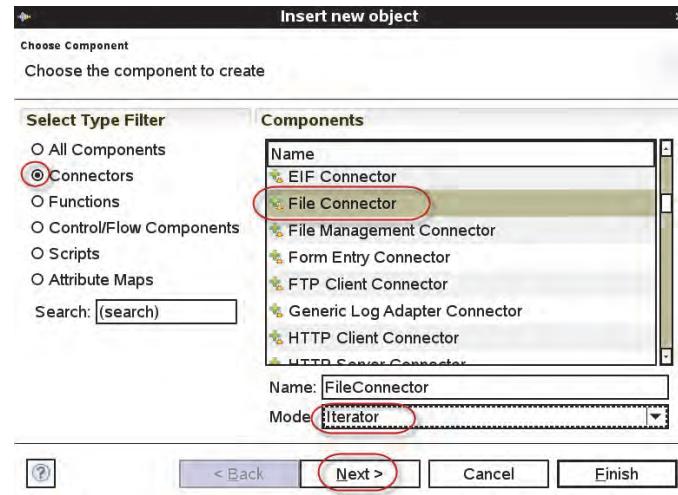
Select **Feed** in the assembly line workspace and click **Add component**.



Selecting a File Connector object

Filter on Connectors and select **File Connector** in the Components list.

- Select Iterator in the connector mode menu and click Next



Tivoli Directory Integrator and DASH basics

48

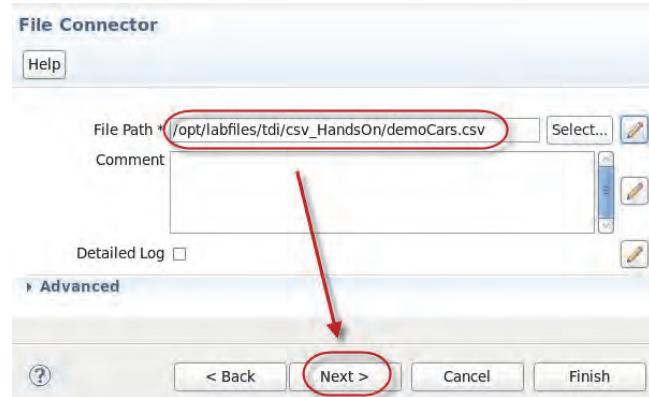
© Copyright IBM Corporation 2016

Selecting a File Connector object

There are several modes supported for the file mode. This example uses Iterator.

Configuring the file connector data source

Enter the fully qualified file name of the data source in the **File Path** field and click **Next**.



Adding a file parser to the connector

- Unstructured data that does not include schema information requires a Parser in the connector configuration
- Required parser parameters vary by parser type
In this example, the CSV field separator is changed to a comma character

Parser Configuration

CSV Parser

Select Parser Help

Field Separator *

Sort fields

Comment

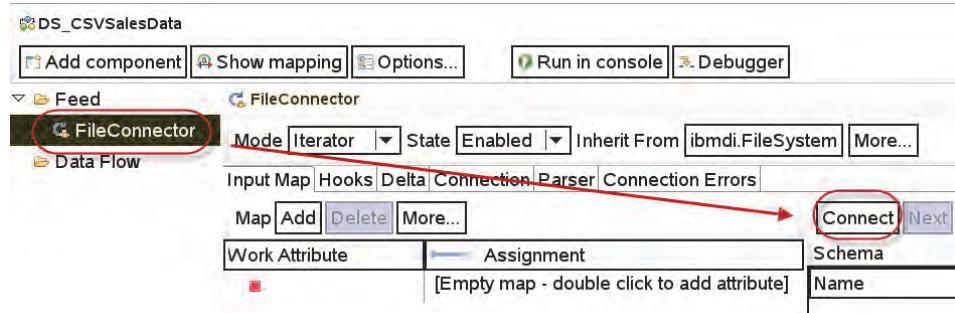
Detailed Log

Adding a file parser to the connector

Note that the default for Comma Separated Values is a semicolon. You must change it to a comma in the exercises to work with the supplied CSV file.

Verifying the file connector operation

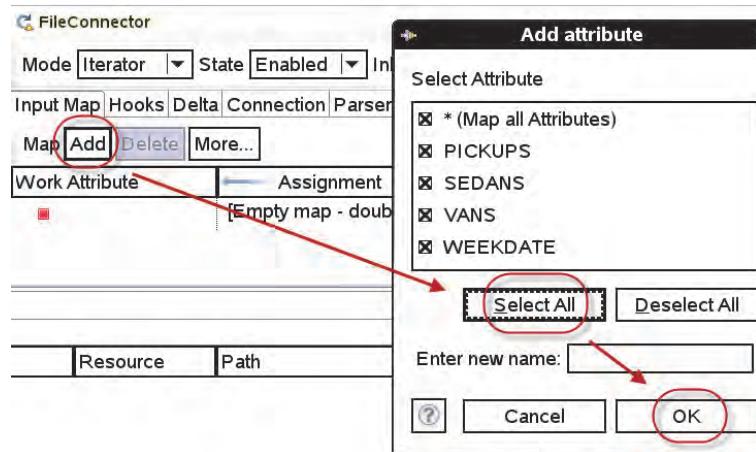
- Click **FileConnector** in the workspace and click **Connect** to read the file data
- Click **Next** to verify that the connector is iterating through the file



Mapping file data to the input map

Select one or more input attributes to map to output attributes

- Click Add and select one or more attributes



Modifying work attribute output values

- The file connector parser reads all source file values as strings, but in this example some data should be integer and some should be in date format

Schema				
Name	Sample Va	Required	Java Class	Native Syntax
PICKUPS	8	Optional	java.lang.String	
SEDANS	10	Optional	java.lang.String	
VANS	5	Optional	java.lang.String	
WEEKDATE	2012-10-0	Optional	java.lang.String	

- To modify, double-click a row in the Work Attribute column and modify the output value with JavaScript code

The screenshot shows the 'Assignment' tab of the 'Map all Attributes' dialog. The 'Work Attribute' column lists 'PICKUPS', 'SEDANS', and 'VANS'. The 'Native Syntax' column shows the corresponding Java code. A red box highlights the 'Native Syntax' column for the 'WEEKDATE' row, which contains 'java.lang.String'. Below the table, the 'FileConnector.PICKUPS' section shows the following JavaScript code:

```
var value = conn.getString("PICKUPS");
value = value.trim();
return java.lang.Integer.parseInt( value );
```

Modifying work attribute output values

All of the values for the names in the schema are string values. You must change the WEEKDATE to a date and the others to numeric values through the JavaScript code.

Deploy the assembly line to the runtime server

- Save the project and drag the project name to the Default entry in the Servers section
- Click OK when presented with an overwrite warning
- The project name entry below the Default server indicates that the project data is cached on the DASH server

Work Attribute	Assignment
PICKUPS	var value = conn.getString("PICKUPS");...
SEDANS	var value = conn.getString("SEDANS");...
VANS	var value = conn.getString("VANS");...
WEEKDATE	conn.WEEKDATE

FileConnector.WEEKDATE

```
var value = conn.getString("WEEKDATE");
value = value.trim();
var ymd = value.split("-");
return new Date(ymd[0],ymd[1]-1,ymd[2],0,0,0,0);
```



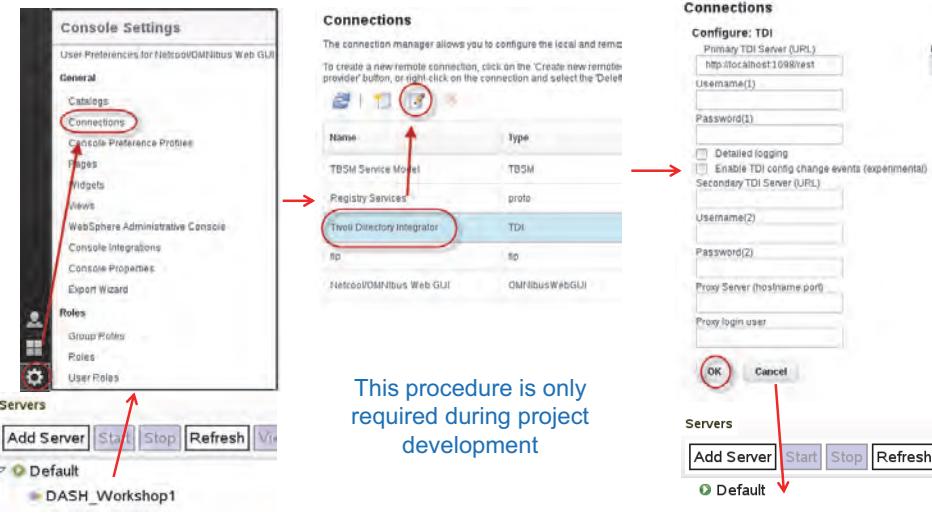
Deploy the assembly line to the runtime server

Process the assignments for the work attributes by supplying the JavaScript code. In the case of the PICKUPS, SEDANS, and VANS, you might have to modify one word in the script.

When you drag and drop the project name DASH_Workshop1 to the Default server, because you already have a project by that name, you are asked to override DASH_Workshop1.

Clearing the Tivoli Directory Integrator cache in the DASH server

Refresh the TDI – DASH cache by opening and saving the TDI connection document in the DASH console, logging off, and logging on



Clearing the Tivoli Directory Integrator cache in the DASH server

Caching is involved with the DASH environment. In order to pick up the most current configuration for DASH_Workshop1, you must select and open the connection and click **OK**. Then you must log off of the console and log back in to the console.

Using the file connector data set with a widget

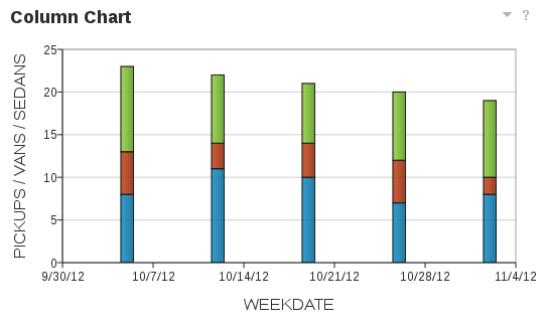
In this example, use the file-based data set in a column chart widget

The screenshot shows the 'Column Chart' configuration interface. At the top, there is a search bar with the placeholder 'Enter a term and press Search to see datasets. To see all, just press Search:' and a red circle around the input field containing 'CSV'. To the right of the input field are 'Search' and 'Cancel' buttons. Below the search bar, a message says 'After searching, select a dataset from the search results below to continue:' followed by 'Provider: Tivoli Directory Integrator - Datasource: Workshop'. A red circle highlights the list item 'CSVSalesData' under 'DS_CVSalesData'. In the main configuration area, there are three sections: 'X axis field' (set to 'WEEKDATE'), 'Y axis value field' (set to 'PICKUPS, VANS, SEDANS'), and 'Group by field' (set to 'None'). Red circles highlight the 'Y axis value field' dropdown and the 'Add' and 'Remove' buttons for the Y-axis fields.

Using the file connector data set with a widget

This example shows how to search and select the CSV data set. As with a previous example using a database, the CSV file has schema entries to select while building the stacked column chart.

Completed chart with file connector data set



Completed chart with file connector data set

The chart with the CSV data looks like the chart previously seen with the DB2 values.

Instructor demonstration

- Creating a UI data provider data set from a CSV file



Exercise 2

Creating a UI data provider data set from a CSV file

Student exercises

Complete Unit 2 Exercise 2 in the exercise guide.

Unit summary

Now that you have completed this unit, you can perform the following tasks:

- Describe the Tivoli Directory Integrator components and architecture
- Use the Tivoli Directory Integrator Configuration Editor to create and deploy custom data sets
- Create Tivoli Directory Integrator assembly lines and connectors that read and serve database data and file data
- Configure widgets to show data that is provided with Tivoli Directory Integrator data sets

Unit 3 Tivoli Directory Integrator and DASH advanced topics

IBM Training



Unit 3 Tivoli Directory Integrator and DASH advanced topics

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit reviews advanced techniques for using Tivoli Directory Integrator to extract and transform data from a variety of back-end data sources and then providing the enhanced data to the DASH server. The student will learn how to reuse common Tivoli Directory Integrator assembly lines to speed up the process of data provider development. You learn how to use helper AssemblyLines to add status data, parameter data, and right-click tasks to data set data. You learn how to create an AssemblyLine that can calculate a metric value from a data source, and how to use a caching connector to support dashboard presentations when a data source is not available.

Objectives

When you complete this chapter, you can perform the following tasks:

- Create Tivoli Directory Integrator UI data providers for various data sources
- Create helper assembly lines to provide parameter selections within data sets
- Create helper assembly lines to provide status information from data source evaluation
- Create helper assembly lines to provide right-click task menus for supported DASH widgets
- Use the cache connector to use offline data for demonstrations
- Create a run-once assembly line to evaluate data source data and provide a key performance indicator or status information

Objectives

The data set contains the data that you want to add to the widget. You can enhance the data by adding features to the data set itself.

The run-once Assembly line will let you process all records in the data set, and then at the end, determine what you want to return from the data set.

Lesson 1 Expanding assembly line functions with helper assembly lines

IBM Training

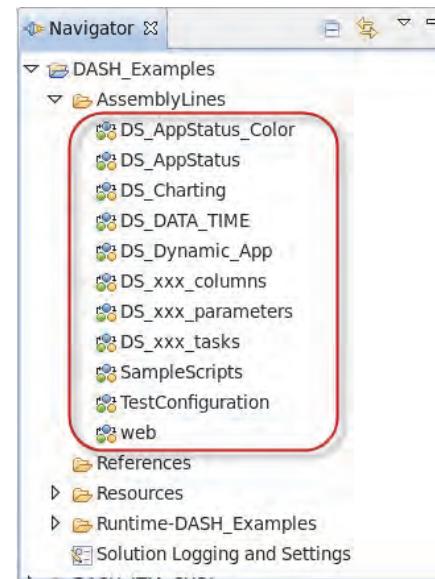
IBM

Lesson 1 Expanding assembly line functions with helper assembly lines

You can add helper assembly lines. When these are associated with a primary assembly line, it will trigger DASH to request the additional information with the original assembly line.

Helper assembly lines

- Helper assembly lines can be added to existing projects to enhance the functions of the UI data provider
- Examples:
 - Add parameter data that provides a choice of what is shown in a dashboard page widget
 - Calculate and provide status data
 - Add right-click menu tasks to a widget
- You can import and adapt template assembly lines to reduce development time

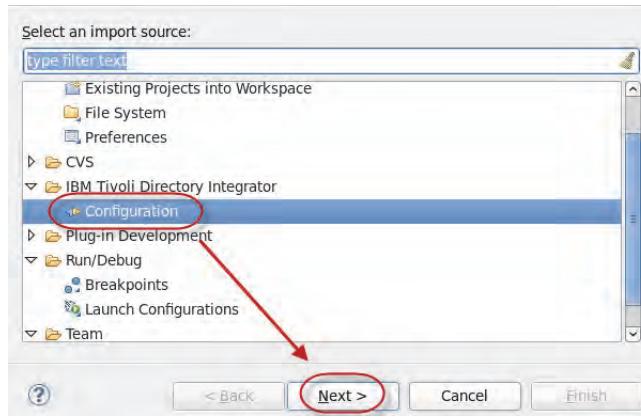


Helper assembly lines

You can evaluate a record in an assembly line, and based on some values, assign a status value to that record.

Importing assembly line templates

- You can save and import assembly line configuration files
 - A set of commonly used assembly line templates are available in your lab environment
- Select the configuration editor File > Import menu
- Leave the default IBM Tivoli Directory Integrator > Configuration selection and click Next



Tivoli Directory Integrator and DASH advanced topics

5

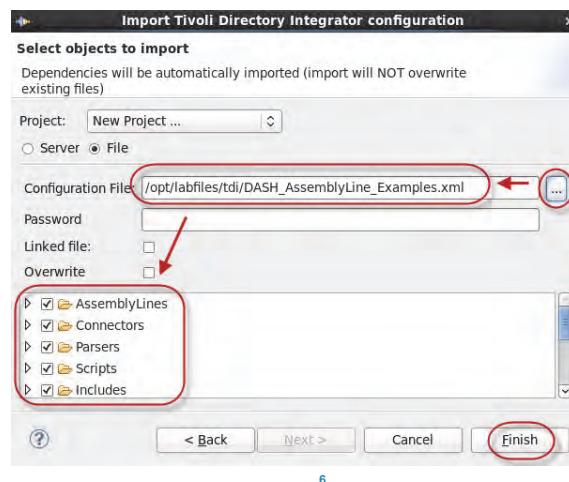
© Copyright IBM Corporation 2016

Importing assembly lines templates

You can import an existing TDI application, modify the application, and then save as a new application. You can import a configuration file.

Selecting the assembly line import scope

- The project configuration file in your lab environment is /opt/labfiles/tdi/DASH_AssemblyLine_Examples.xml on the dash151 virtual image
 - Select all available project components and click Finish



Tivoli Directory Integrator and DASH advanced topics

6

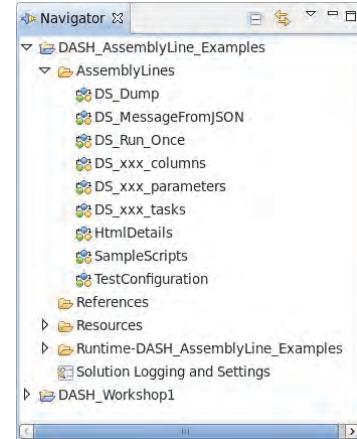
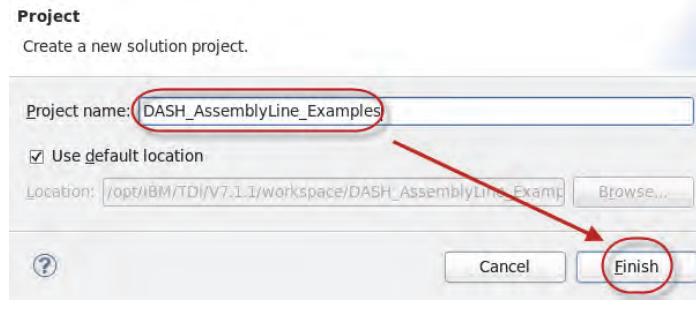
© Copyright IBM Corporation 2016

Selecting the assembly lines import scope

You can import AssemblyLines into an existing or a new project.

Project view after importing the configuration file

- Enter DASH_AssemblyLine_Examples in the Project name field and click Finish
 - You can copy assembly lines from one project to another with standard editor copy and paste methods
- Expand the AssemblyLines folder in the configuration editor navigator to see the assembly line list



Project view after importing the configuration file

In the directory listed in the project name, you will see many examples made available to this course.

Lesson 2 Creating a parameter helper assembly line

IBM Training



Lesson 2 Creating a parameter helper assembly line

Example: Add location parameters to car sales data

- Scenario: Weekly car sales are gathered and stored by location
 - The VP of Sales wants to be able to see sales based on the location
 - The original sales database is updated with new data that includes the location
- You need to create a location-based parameter-based query in the sales database UI data provider and make the parameter selection available to the dashboard user



Table		
WEEKDATE	TOTAL	LOCATION
Oct 5, 2012 1:00:00	23	NC
Oct 12, 2012 1:00:00	22	NC
Oct 19, 2012 1:00:00	21	NC
Oct 26, 2012 1:00:00	20	NC
Nov 2, 2012 1:00:00	19	NC

Tivoli Directory Integrator and DASH advanced topics

9

© Copyright IBM Corporation 2016

Example: Add location parameters to car sales data

This example uses the Car Sales database again. Now location information is being added to the records. The dataset that is created will show the sales for a particular location. Hot spots are added to the map so that you select only that location where the hot spot is, and only see sales data from that state.

Adding parameters to a data set

- To add parameters to a data set:
 - Create the primary assembly line with the name form DS_ALName
 - Create a helper assembly line with the name form DS_ALName_parameters
- If the DS_ALName values do not match in both names, the DASH server does not recognize the helper assembly line

The screenshot shows the DASH interface with the following details:

- Project Structure:** DASH_Test_Project > AssemblyLines > DS_TestDBSales.
- Dataset Selection:** Selected Dataset: Tivoli Directory Integrator > Test_Project > TestDBSales. DS_TestDBSales. Data Type: table, No Automatic Refresh, Local Data Provider.
- Required Settings:** *Required Settings (highlighted with a red box). Error message: **No visualization attribute found for mapping to dataset columns.**
- Optional Settings:** Optional Settings (highlighted with a red box).
- Configure Mandatory Dataset Parameters:** A dropdown menu labeled "Select Location" with "Store location" and "California" selected (highlighted with a red box).
- Bottom Labels:** "DASH widget quick edit with only the primary assembly line defined" and "DASH widget quick edit with _parameters helper assembly line".

The Helper AssemblyLine must match the original primary AssemblyLine name, plus “_parameters” appended to the name. Without that “_parameters” appended, it won’t be recognized.

Updated database

- For this example, the sales database is updated with a new table that includes model and location data for the corresponding sales numbers.
- You want to be able to specify a default location when you configure a widget
 - For example, initially show sales numbers for the California offices, but include a way to select and show North Carolina values in the same widget

LOCATION	WEEKDATE	MAJORTYPE	MINORTYPE	SALES	LABEL
CA	2012-9-05	SEDAN	HYBRID	8	Hybrid Sedan, CA, 2012-10-05
CA	2012-10-05	SEDAN	SPORT	4	Sports Sedan, CA, 2012-10-05
CA	2012-10-12	SEDAN	HYBRID	2	Hybrid Sedan, CA, 2012-10-12
CA	2012-10-12	SEDAN	SPORT	3	Sports Sedan, CA, 2012-10-12
NC	2012-10-19	SEDAN	HYBRID	4	Hybrid Sedan, NC, 2012-10-19
NC	2012-10-19	SEDAN	SPORT	4	Sports Sedan, NC, 2012-10-19
...					

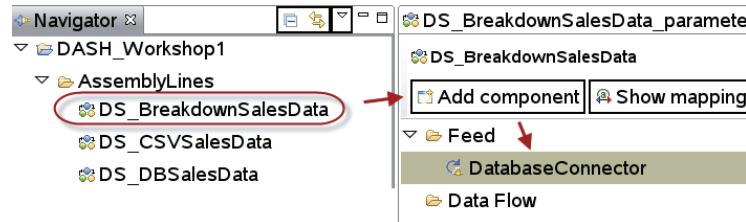
Updated database

Location is part of the database schema. Two states, NC and CA, are the values for the location.

Configuring the primary assembly line

First, create the primary assembly line that iterates through the new sales data table

- Create an assembly line in the DASH_Workshop1 project called DS_BreakdownSalesData
- Next, add a database connector component to the Feed section of the assembly line

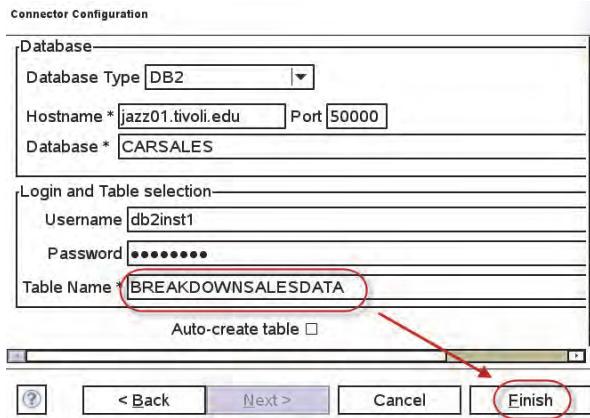


A database connector is added to DS_BreakdownSalesData in the Feed section.

Configuring the database connector

Configure a DB2 connector in the assembly line feed section

- Use the updated table, BREAKDOWNSALES DATA, in the Table Name field
- Map all the database columns to the assembly line work entries



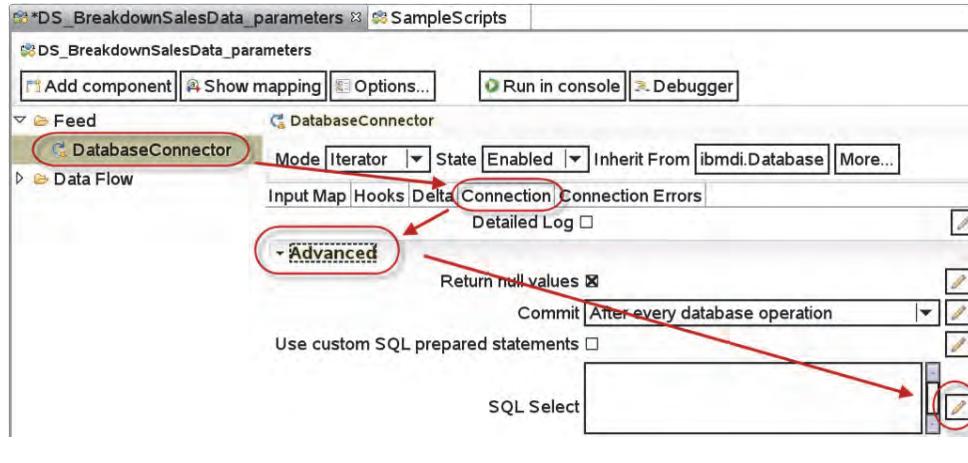
Configuring the database connector

When the database type DB2 is selected, the window changes and expects other database related values to be provided. The CARSales database was entered. To the side of the Table Name field, a click displays the names of the tables in the CARSales database.

Creating an SQL select statement in the database connector

Modify the connector to sort the sales table data by location and sum all sales by date

- Select DatabaseConnector in the Feed section, and select the Connection tab
- Expand the Advanced section and click the edit icon in the SQL Select section



Creating an SQL select statement in the database connector

The mapping is a similar to what was performed previously. But this time, it will be configured with a SQL Select statement. When it is triggered, it will select a subset of the original data.

Using a script for the SQL select configuration

- A sample SQL query script is provided in the file **/opt/labfiles/tdi/parameters_HandsOn/parameter_select.js**
- Select Advanced (JavaScript) to open the script field
- Open **parameter_select.js** in a text editor, copy the contents, paste into the SQL Select field of the connector configuration, and click **OK**

```
var myvalue = "NC";
if (typeof(task) != "undefined" && task.getOpEntry() != null) {
  loc = task.getOpEntry().getString("LOCATION");
  if (loc && loc.length)
    myvalue = loc;
}
var sql = "select WEEKDATE, LOCATION, SUM(SALES) as TOTAL from BREAKDOWNSALESDATA where
LOCATION = '" + myvalue + "' GROUP BY WEEKDATE, LOCATION";
return sql;
```

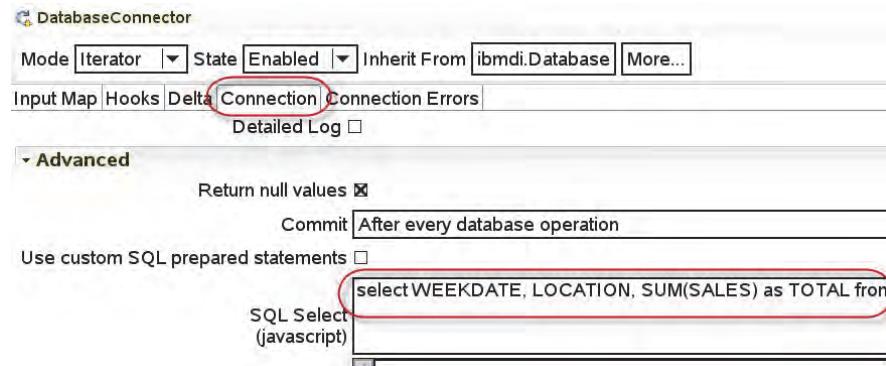
Using a script for the SQL select configuration

The variable is set to a location for default purposes. It could be set to something invalid, but this example is using NC. This SQL statement filters the location selected. A variable named sql is created and returned. The LOCATION is dynamic where it is equal to myvalue.

SQL select after saving script

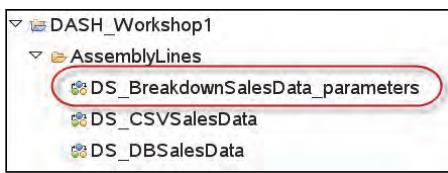
Click **OK** to save the SQL Select configuration

- Only the final SQL statement is shown, with the parameter variable name highlighted in color



Adding a helper assembly line to a project

- Right-click **DS_xxx_parameters** in the imported examples project and select the **Copy** menu
- Right-click the **AssemblyLines** folder in the target project and select the **Paste** menu
- Right-click the copied assembly line and select the **Rename** menu
- Enter **DS_BreakdownSalesData_parameters**
 - The **_parameters** suffix is a special indicator to the DASH server



Adding a helper assembly lines to a project

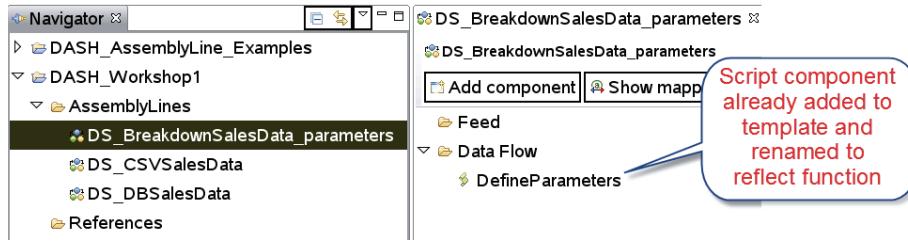
The primary assembly line is created. Now the helper assembly line is created.

To start, a copy is made of DS_XXX_parameters, and pasted into DASH_Workshop1 > AssemblyLines. As stated previously with another assembly line, it must be renamed to match the name of the primary assembly line with “_parameters” appended to it.

Modifying an assembly line template

The parameters assembly line must define and provide the values and labels for the data set parameters

- The template includes a sample **DefineParameters** script
- Modify the script to match the car sales by location example



Modifying an assembly lines template

There is no feed because the values in the primary assembly line will become input variables to the helper assembly line. The example has a define parameters that was copied to the Data Flow.

Configure the DefineParameters script component

- A sample parameters script is included in the assembly line template
- Click DefineParameters in the Data Flow section of the assembly line to see the sample script in the workspace

```
var parameterList = new java.util.ArrayList();
// This is how to define a string parameter
var param = system.newEntry();
param.type = "string"
param.id = "node"
param.label = "Server selection"
param.description = "List jobs for the specified server"
param.hidden = false
param.required = false
parameterList.add(param);
```

Configure the DefineParameters script component

The contents of the DefineParameters script is displayed.

Modifying the DefineParameters template script

```
var parameterList = new java.util.ArrayList();  
  
// This is how to define a string parameter  
var param = system.newEntry();  
param.type = "string"  
param.id = "node"  
param.label = "Server selection"  
param.description = "List jobs for the specified server"  
param.hidden = false  
param.required = false  
parameterList.add(param);  
  
// This is how to define a parameter where the user picks  
var param = system.newEntry();  
param.type = "map"  
param.id = "Selection"  
param.label = "Select value type"  
param.description = "Description of the parameter"  
param.hidden = false  
param.required = true  
  
// Define the values you want the user to pick between  
var values = new java.util.ArrayList();  
param.values = values;  
  
var v1 = system.newEntry();  
v1.label = "First value"  
v1.value = "Value1"  
values.add(v1);  
  
v1 = system.newEntry();  
v1.label = "Second value"  
v1.value = "Value2"  
values.add(v1);
```

Delete the section that defines a string parameter. It is not used in this example.

Change the following values:
param.id = "LOCATION"
param.label = "Select Location"
param.description = "Store location"

Change the following values:
v1.label = "California"
v1.value = "CA"

Change the following values:
v1.label = "North Carolina"
v1.value = "NC"

Modifying the DefineParameters template script

This slide shows the changes bounded by a red oval that you will make to the script in your lab exercises. These changes are made to accommodate the NC and CA hot spot selections. The bottom two parts create entries for an array.

Final Script

- Final script after modifications
 - The script creates a parameter that is called LOCATION and assigns a label and description
 - The script then creates an array to store the parameter selection values
 - The script then adds two location values to the array, corresponding to the car store locations
 - These values are shown in the UI data provider as parameter selections
- Save the project, add the project to the runtime server, and recycle the Tivoli Directory Integrator connection cache in the DASH server

```
var parameterList = new java.util.ArrayList();

// This is how to define a parameter where the user picks
var param = system.newEntry();
param.type = "map"
param.id = "LOCATION"
param.label = "Select Location"
param.description = "Store location"
param.hidden = false
param.required = true

// Define the values you want the user to pick between
var values = new java.util.ArrayList();
param.values = values;

var v1 = system.newEntry();
v1.label = "California"
v1.value = "CA"
values.add(v1);

v1 = system.newEntry();
v1.label = "North Carolina"
v1.value = "NC"
values.add(v1);

// add param to the paramlist
parameterList.add(param);

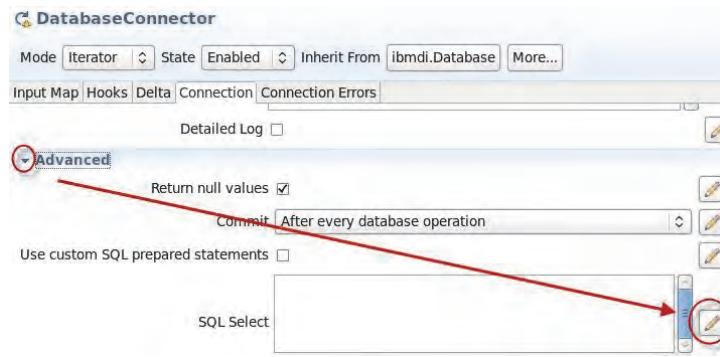
// finally return the param list
work.result = parameterList;
```

Final Script

The finished script looks like the example in the slide. The selection in the parameter section will determine which location entries will be included.

Modifying the database connection to use parameter selections

- The primary assembly line database connector must be modified to accommodate a parameter selection from a dashboard widget
- For example, how to change the retrieved data set when a user selects North Carolina or California in a dashboard page
- Create an SQL Select statement in the Advanced settings for the database connector

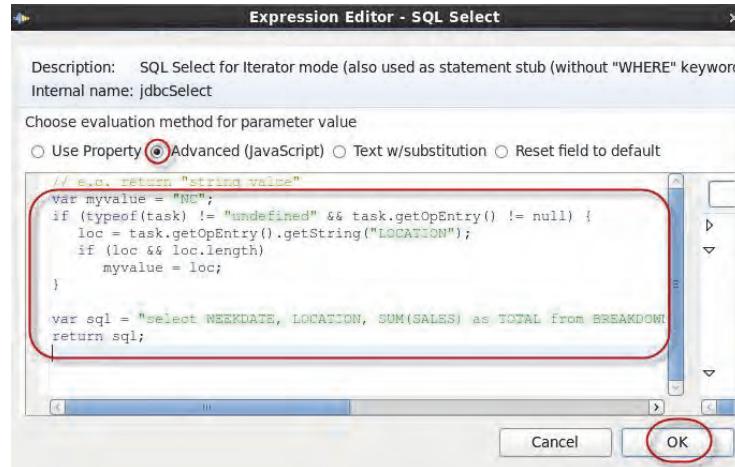


Modifying the database connection to use parameter selections

Next, you work with the SQL Select statements. You click the Advanced arrow to display the SQL area.

SQL Select configuration

Select **Advanced (JavaScript)** and add a script that uses the value of LOCATION, passed in by the helper assembly line, to create the data filter



SQL Select configuration

Advanced (Java Script) is selected in the example.

Using the parameterized UI data provider in a dashboard

In this example, create a page called “Sales by location”

- The page contains a US map image widget, two hotspot widgets, and a table widget
- Configure the hotspot widgets to send location data to the table widget when selected
- Configure the table widget to show sales data, by location
- Configure the table widget to subscribe to NodeClickedOn events (default is off)



Table			
Name	Type	Status	Description
Item 0	Test Item	OK	Description for Item 0
Item 1	Test Item	Service	Description for Item 1
Item 2	Test Item	Critical	Description for Item 2
Item 3	Test Item	Unknown	Description for Item 3
Item 4	Test Item	Warning	Description for Item 4
Item 5	Test Item	Inactive	Description for Item 5
Item 6	Test Item	Information	Description for Item 6
Item 7	Test Item	Unknown	Description for Item 7
Item 8	Test Item	Unrecoverable	Description for Item 8

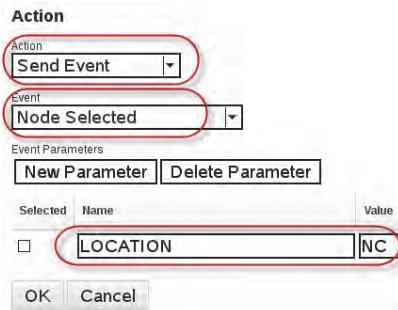
Using the parameterized UI data provider in a dashboard

The assembly line is saved. The project is added to the runtime server. Now that project dataset can be accessed by a DASH page.

Hot spot widgets can be used to generate a NodeClickOn event when selected. That event includes the location value. The table is configured to show data for one location initially. By selecting a hot spot, the table will trigger the assembly line and will update the data in the table with the hot spot location.

Configuring the hotspot widgets

- Edit the North Carolina hotspot widget to send a location context event when selected
 - Select Send Event in the Action menu
 - Select Node Selected (NodeClickedOn) in the Event menu
 - Enter LOCATION in the parameter Name field and NC in the Value field
- Repeat the process for the California hotspot widget, but set the LOCATION value to CA



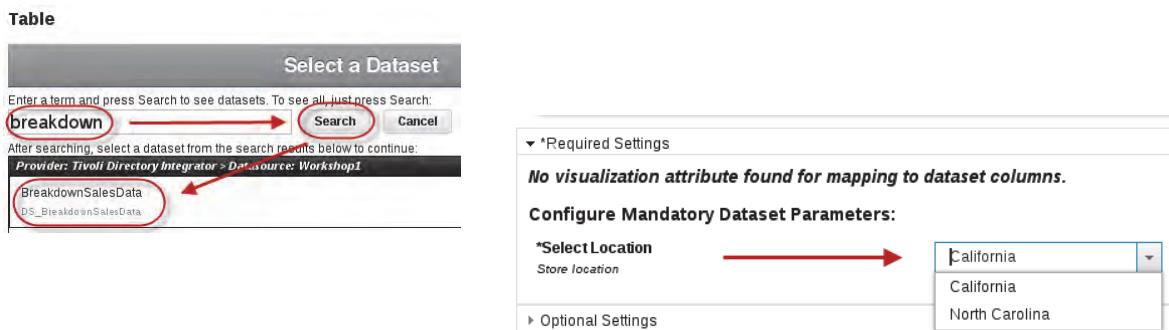
Configuring the hotspot widgets

From DASH, you configure the hot spots with the information bounded by the red ovals in the slide.

Configuring the table widget

You want to show data for a specific location in the table widget

- Edit the table widget:
 - Enter breakdown in the Search field and select the BreakdownSalesData data set
 - Select either location in the Select Location menu
- The selected location is the default value that is initially shown in the table

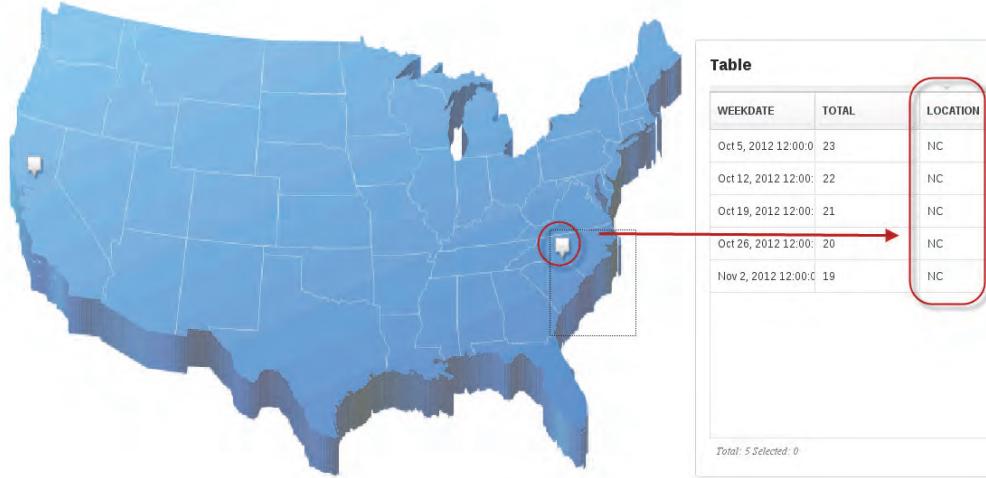


Configuring the table widget

This begins the table widget configuration. A default LOCATION is established.

Testing the updated Sales by Location page

- In this example, the table widget was configured to initially show CA data
- Clicking the North Carolina hotspot triggers the table widget to update with NC data



Testing the updated Sales by Location page

Clicking the NC hot spot shows NC data.

Instructor demonstration

- Using a helper assembly line to add parameters to a data set



Exercise 1

Using a helper assembly line to add parameters to a data set

Student exercises

Complete Unit 3 Exercise 1 in the exercises guide.

Lesson 3 Creating a status column helper assembly line

IBM Training

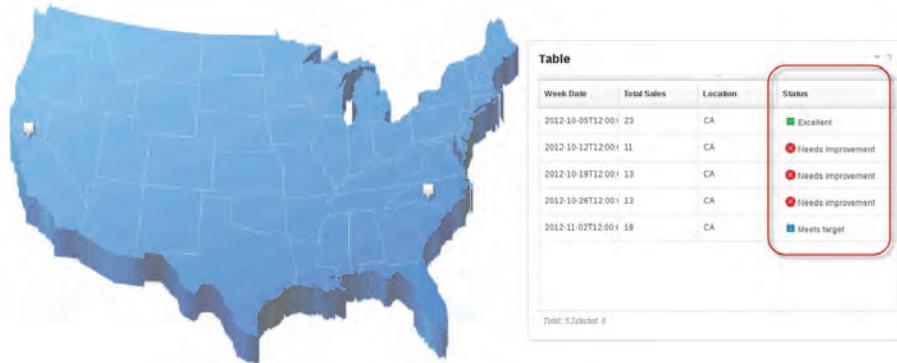
IBM

Lesson 3 Creating a status column helper assembly line

A status column helper is added to indicate the quality or deficiency of sales, represented by a status column.

Adding data set columns to show status in a data set

- Database data can contain values that can be used to assign a status value, but the value is not in a format that is used by a widget. In this example, you learn how to supplement a data set data with corresponding status values
- Scenario: You need to use threshold values to visually flag sales numbers as Excellent, Needs Improvement, or Meets target
 - Add a column that includes a status attribute based on a sales threshold



Adding data set columns to show status in a data set

A status column is added by evaluating the data.

Columns assembly line overview

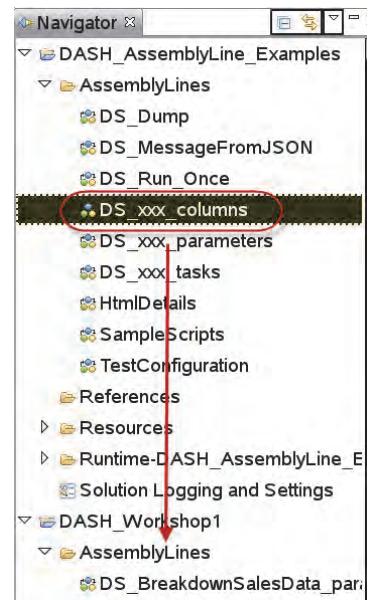
- The DS_xxx_columns helper assembly line template provides a set of template scripts to manually define data set columns
 - The _columns suffix is a special indicator to the DASH server
 - Even though you need to add only one new column to show status, you need to define all columns in the data set
- To define status values for a data set, you must modify the _columns helper assembly line and the primary assembly line data flow:
 - Copy the DS_xxx_columns assembly line template to the target project AssemblyLines folder (for example, DASH_Workshop1 > AssemblyLines)
 - Rename the copied template to match the primary assembly line name (for example, DS_BreakdownSalesData_columns)
 - Modify the Define columns script in the new assembly line to include column definitions for all columns in the primary assembly line
 - Modify the StatusUtility script properties in the new assembly line to set the script to be a global resource
 - Add a script to the primary assembly line Data Flow section to evaluate threshold values and assign a corresponding status value

Columns assembly lines overview

The columns helper assembly line will define all of the columns included in the dataset plus it will add a status column. That status column is updated based on some criteria.

Copying the _columns template

- Right-click the **DS_xxx_columns** template in the **DASH_AssemblyLine_Examples** assembly line list and select **Copy**
- Right-click the **AssemblyLines** folder in the **DASH_Workshop1** project and select **Paste**

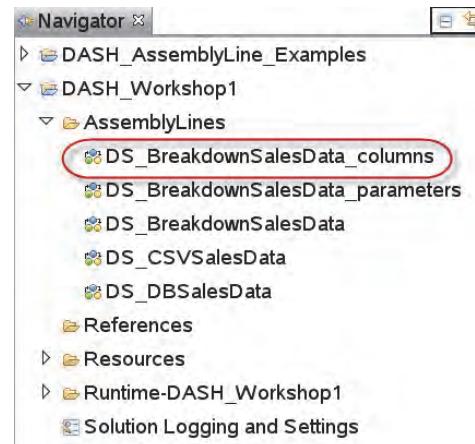


Copying the _columns template

The slide shows how the DS_xxx_columns is copied into the primary project assembly lines. T

Renaming the new assembly line

- Right-click the new assembly line and select **Rename**
- Change the name to include the **_columns** suffix and the name of the primary assembly line
- For example, **DS_BreakdownSalesData_columns**

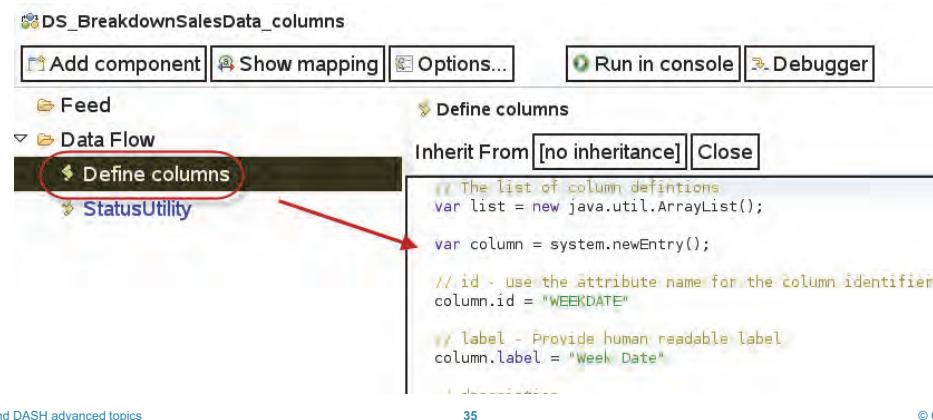


Renaming the new assembly lines

The DS_xxx_columns assembly line is renamed to match the BreakdownSalesData name.

Defining each column in the helper assembly line

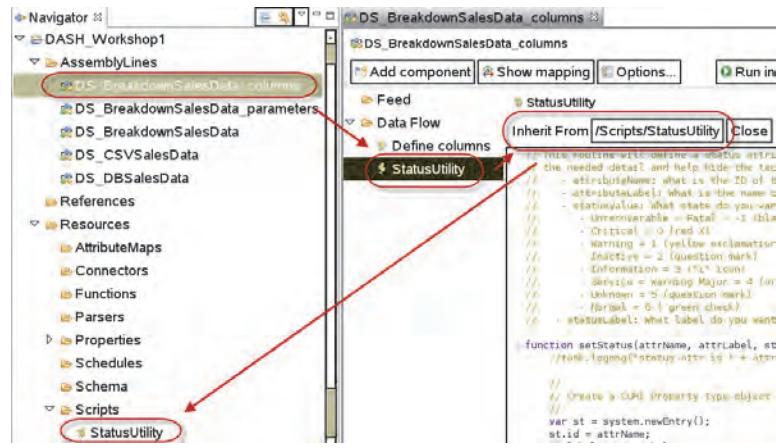
- Open the DS_BreakdownSalesData_columns assembly line and click Define columns in the Data Flow section
 - The included template script shows an example definition for one column
- Modify the script to define columns for the WEEKDATE, TOTAL, LOCATION, and STATUS columns
 - A sample script is included in the /opt/labfiles/tdi/columns_Hands_On/BreakdownSalesData_columns.js file



The StatusUtility script

The _columns template includes a second script, called StatusUtility

- The script is called from the primary assembly line to calculate status values
- The script in the Data Flow is inherited from the file in Resources > Scripts



Tivoli Directory Integrator and DASH advanced topics

36

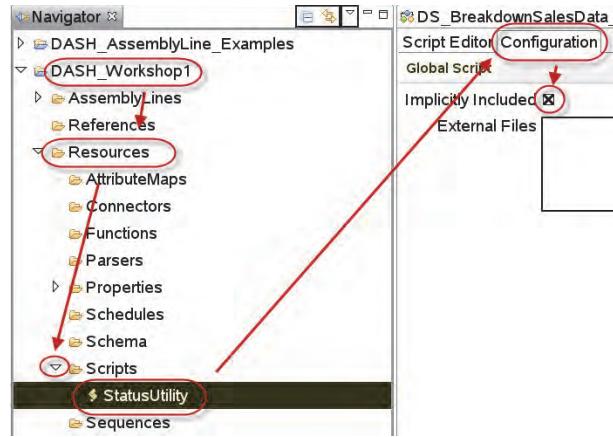
© Copyright IBM Corporation 2016

The StatusUtility script

The statusUtility script is moved to the Scripts folder in the DASH_Workshop1 column. The script is made a global resource.

Setting the StatusUtility script scope

- Objects in the project Resources folder are configured with local or global scope, relative to the project assembly lines
- Double-click Resources > Scripts > StatusUtility, click Configuration, and select Implicity Included to set the scope to Global



Tivoli Directory Integrator and DASH advanced topics

37

© Copyright IBM Corporation 2016

Setting the StatusUtility script scope

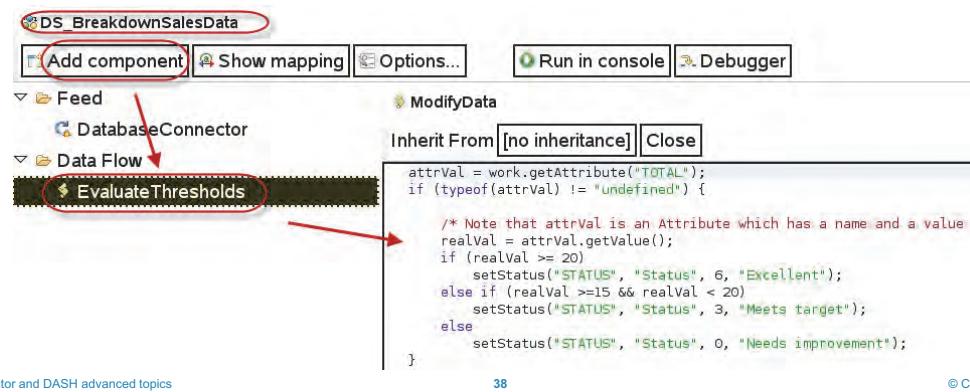
The StatusUtility script is opened. The Implicitly included box is checked, making the script a global resource.

Evaluating status thresholds

Create a threshold evaluation script in the primary assembly line

- The script calls the setStatus function in the _columns assembly line, which is why the script scope must be set to global
- In this example, the thresholds are set to:

≥ 20	:	Excellent
$<20 \& \geq 15$:	Marginal
< 15	:	Bad



Evaluating status thresholds

In the EvaluateThresholds script, the status criteria is applied, based on the numbers of sales: Excellent, Meets target, or Needs improvement.

Testing the _column helper assembly line

- Edit the Sales by Location page, replacing the previously created table widget
- Configure the new table widget to use the DS_BreakdownSales assembly line
 - The parameter selection from the _parameters assembly line is still functional
 - Edit the table widget Events properties to subscribe to NodeClickedOn events



Testing the _column helper assembly lines

Save and update the Runtime Server with this new assembly line. Configure the new table widget with this updated dataset.

Instructor demonstration

- Using a helper assembly line to add status values to a data set



Exercise 2

Using a helper assembly line to add status values to a data set

Student exercises

Complete Unit 3 Exercise 2 in the exercises guide.

Lesson 4 Creating a tasks helper assembly line

IBM Training

IBM

Lesson 4 Creating a tasks helper assembly line

This lesson adds a helper assembly line that uses a right-click helper assembly line.

Adding task menus to a data set

- There might be scripts, application consoles, or related information that you need to make available to console users
 - You can add a DS_<AssemblyLineName>_tasks helper assembly line to provide context menus or start task scripts or open informational pages
 - The _tasks suffix is a special indicator to the DASH server
- Scenario: When a problem is found with sales numbers, executives want to be able to easily find the sales manager with responsibility for the sales. You need to add a widget right-click task that opens a page to show the resource owner contact information



Table			
Week Date	Total Sales	Location	Status
2012-10-05T12:00:01	23	CA	Excellent
2012-10-05T12:00:01	13	CA	Needs improvement
2012-10-26T12:00:01	13	CA	Needs improvement
2012-11-02T12:00:01	18	CA	Meets target

How to create a _tasks helper assembly line

To create a _tasks assembly line, complete the following tasks:

- Copy the DS_xxx_tasks assembly line template into your project AssemblyLines folder
- Rename the new assembly line to match the primary assembly line name (for example, DS_BreakdownSalesData_tasks)
- Modify the included tasks script in the Data Flow section of the new assembly line as needed
- Update the project in the runtime server

How to create a _tasks helper assembly line

The process to create a _tasks helper assembly line is basically the same as with the other helper assembly lines.

Creating the task helper assembly line

- Right-click the DS_xxx_tasks template in the DASH_AssemblyLine_Examples list and select Copy
- Right-click the AssemblyLines folder in the DASH_Workshop1 project and select Paste

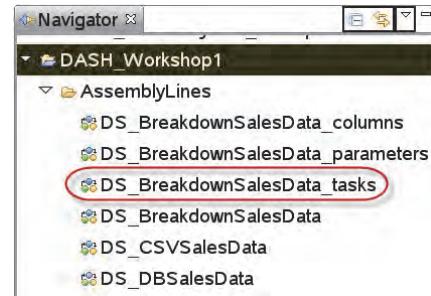


Creating the task helper assembly line

Right click and copy the example tasks helper assembly line. Paste this tasks helper assembly line into the assembly lines folder.

Renaming the helper assembly line

- Right-click the new assembly line and select Rename
- Change the name to include the _tasks suffix and the name of the primary assembly line
- For example, DS_BreakdownSalesData_tasks

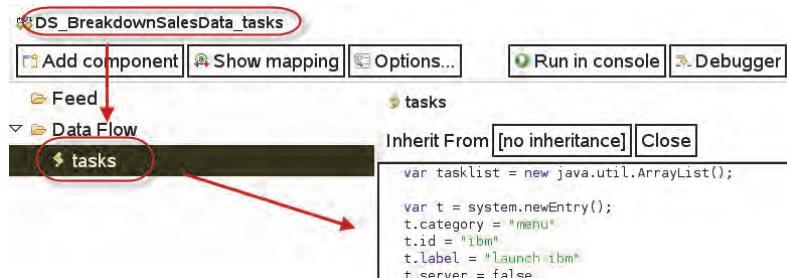


Renaming the helper assembly line

The copied tasks helper assembly must be renamed to match the primary assembly line name and become part of that set of assembly lines.

Reviewing the sample tasks script

- Open the sample task script in the new assembly line
 - Double-click the new assembly line in the Navigator and click tasks in the Data Flow section
- The sample script includes three types of launch configurations
 - Simple URL launch
 - Open an internal DASH page
 - Use selected context information to dynamically create a URL launch
 - In this example, you want to open an internal DASH page



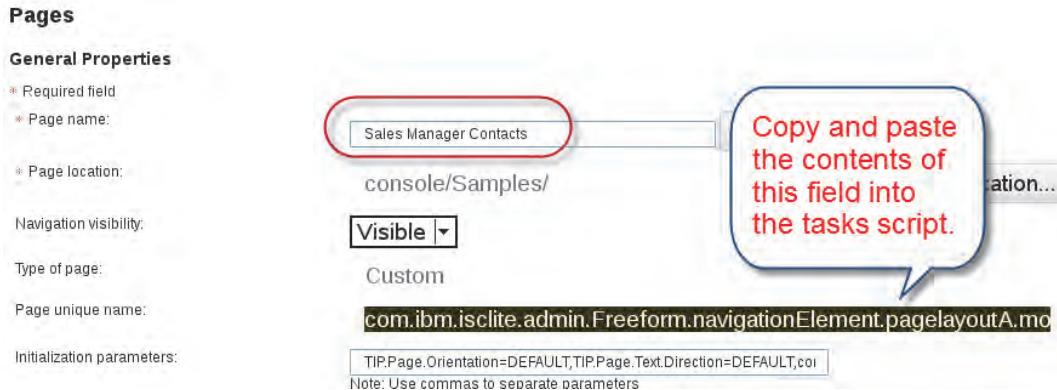
Reviewing the sample tasks script

Examine the script. Tasks can open another page or launch another URL. Our example opens another page that contains sample contact information for the Cale Sales team.

Looking up an internal page ID

Before you modify the tasks script, you need to look up the page ID for the launch target page

- Open the target page from the Console Settings > Pages task
- Copy and paste the Page unique name value into the tasks script



Looking up an internal page ID

The Page unique name can be very long. Be sure when you copy it that you copy the entire page name. Paste this page unique name into the tasks script.

Modifying the tasks script

- Modify the section that is documented for opening an internal page
- Change the following parameters:

```
t.pageId = "<Paste the Page unique name value between the quotations>"  
t.label = "Sales Manager Contacts"
```

- Save the project and update the runtime server

```
// This is an example task that will navigate the user to another page in DASH. All you  
// need to do is bring up the Pages task in DASH, find the page you want to navigate to,  
// then click on it to see its details. On this details page you will see a property  
// called "Page Unique Name". That would be the value you pass in for pageId.  
var t = system.newEntry();  
t.category = "menu";  
t.pageId = "com.ibm.isclite.admin.Freeform.navigationElement.pageLayoutA.modified.z-pcsh";  
t.label = "Sales Manager Contacts";  
t.server = false;  
t.type = "page";  
tasklist.add(t);
```

Modifying the tasks script

Note the parameters and where the Page unique name is copied: t.pageID.

The page unique name is case sensitive and can not have spaces in it.

Testing the _tasks helper assembly line

- Recycle the DASH server connection cache
- Open the **Sales By Location** page, right-click a row of the table, and select the new task

Table			
Week Date	Total Sales	Location	Status
2012-10-05T12:00:00	23	CA	■ Excellent
2012-10-12T12:00:00	18	CA	● Needs improvement
2012-10-19T12:00:00	13	CA	● Needs improvement
2012-10-26T12:00:00	18	CA	■ Meets target

US Sales Manager Contact Information

California

Sales Manager: Joan Smith
Mobile: (555) 555-5555
Office: (555) 555-5555
Email: jsmith@acmecars.com

Assistant Sales Manager: Earl Hammer
Mobile: (555) 555-5555
Office: (555) 555-5555
Email: ehammer@acmecars.com

North Carolina

Sales Manager: William Gibbons
Mobile: (555) 555-5555
Office: (555) 555-5555
Email: hgibbons@acmecars.com

Assistant Sales Manager: Eileen Ripley
Mobile: (555) 555-5555
Office: (555) 555-5555
Email: eripley@acmecars.com

Testing the _tasks helper assembly lines

Add the dataset to the runtime server and refresh the cache which is important.

When you right click an entry in the table, select the Sales Manager Contacts. The selection opens the contact information.

Exercise 3 Using a helper assembly line to add menu tasks to a data set

Complete Unit 3 Exercise 3 in the exercise guide.

Lesson 5 Creating a run-once assembly line

IBM Training

IBM

Lesson 5 Creating a run-once assembly line

Run-once Assembly Line

- Run-once assembly lines are useful when you always need to return a single value, regardless of the data source
 - Use when the data source might return no data
 - Use when you need to sum or aggregate data from a set of metrics
- Scenario: A customer uses SmartCloud Monitoring agents on several hypervisor systems. You need to configure an assembly line to return a count of the number of hypervisors that are over or under used
 - Query for hypervisors, sum up data across them
- Solution: Query the Tivoli Enterprise Monitoring Server server to find all critical or warning events that are issued for monitored systems and count the number of each type of message
- Scenario: A customer needs to extract a metric from a CSV file
- Solution: Parse a CSV, looking for a marker in the data to designate the end of the report data. Return the data from the line that is found just before the end marker.

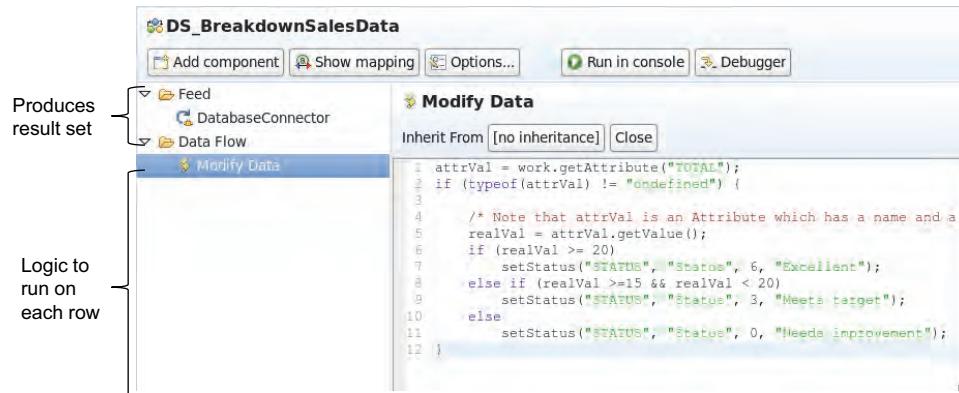
Run Once assembly line

All of the assembly lines that you created read through the data lines one at a time. The data was processed and information was returned for each line of data that was read. This assembly line will read all the way through the data, and then determine what you want to return.

Your exercises will do the last scenario, where you look through a CSV file for a specific set of numbers. Once these numbers are located, then data is returned from the lines ready prior to that set of numbers.

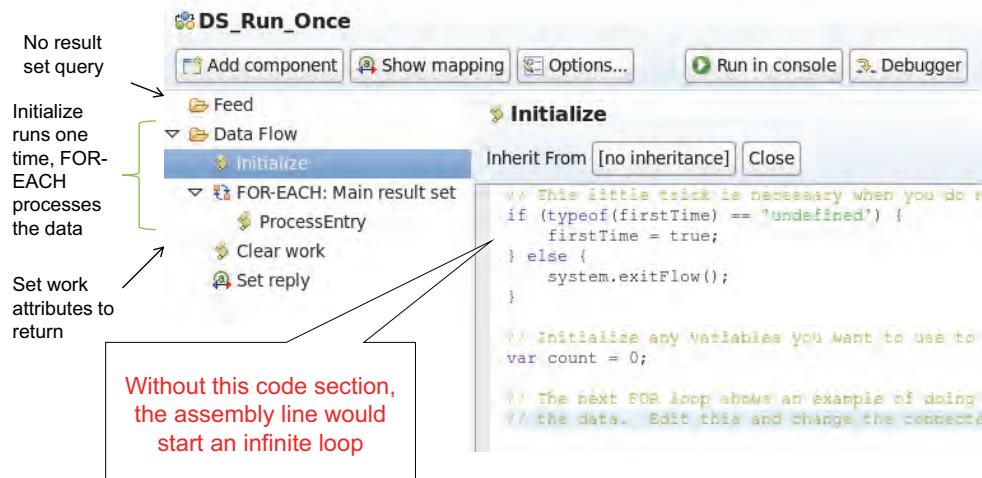
Typical assembly line review

- A typical AL has a query in the Feed section with some type of connector
- The returned data set is processed, one row at a time, in the Data Flow section



Run-once assembly line review

- A run-once AL has no result set query (does not return a list of records)
- The Initialize section runs one time and the FOR-EACH loop processes the data from the configured data source and returns the found data attributes



Tivoli Directory Integrator and DASH advanced topics

70

© Copyright IBM Corporation 2016

Run-once assembly line review

A feed is not configured for this run-once assembly line, because we do not want it to process each record individually and do a function. Instead, all records will be processed before action takes place.

The For Each component is an assembly line loop. Within the connector of the For Each component, the processEntry script performs the function. The Clear work data flow will clear the object because that is not the data that you want to return. The Set reply is the data that is returned.

Example: Parsing a file and extracting metrics

- A customer has a process that periodically appends data rows to a CSV file
- Each row contains four fields, with each field showing a calculated metric
- When all useful data is added to the file, the process appends a row of zeros (0,0,0,0)
 - Values that are appended after the zero row are part of the next processing period
- You need to create and configure a run-once assembly line that returns the metrics for the column 1 and column 2 values in the final nonzero row of data
- For example, assume that the file contains the following data:
 - 1, 2, 3, 4
 - 4, 3, 3, 2
 - 0, 0, 0, 0
 - 3, 4, 5, 2
- The assembly line should return COL1=4 and COL2=3

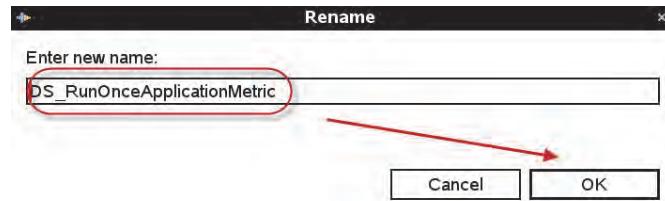
Example: Parsing a file and extracting metrics

The row of all zeros is being used as an end of file marker to end the record reading process. Once the zeros are found, it is the previous row that we are interested in. In the example on the slide, we are only returning columns 1 which has a value of 4, and column 2 which has a value of 3.

Using the run-once assembly line template



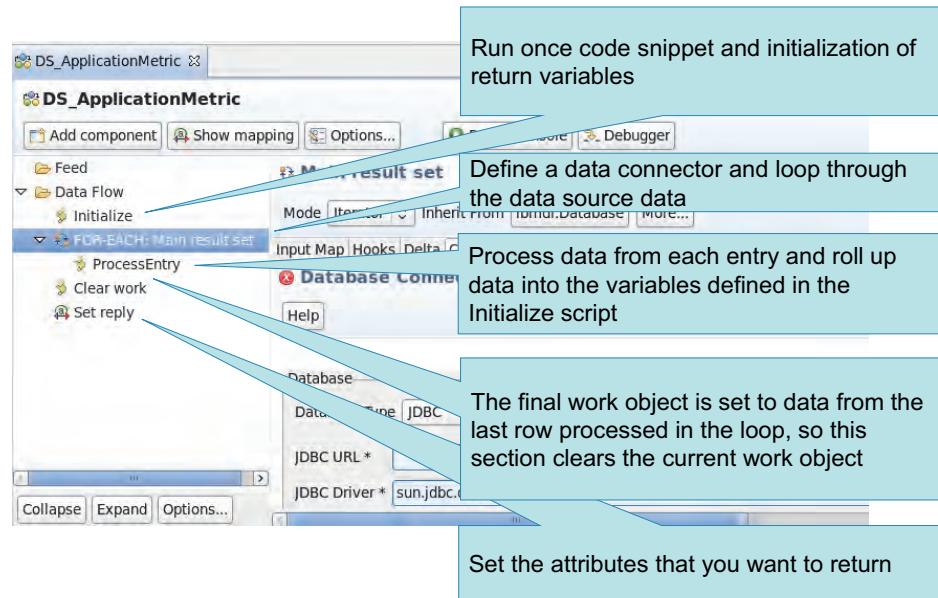
- Copy the DS_Run_Once assembly line from the DASH_AssemblyLine_Examples project
- Paste it into your DASH_Workshop1 project
 - Rename it to DS_RunOnceApplicationMetric



Using the run-once assembly line template

The DS_Run_Once assembly line is copied to the DASH_Workshop1 project.

DS_Run_Once assembly line processing flow review

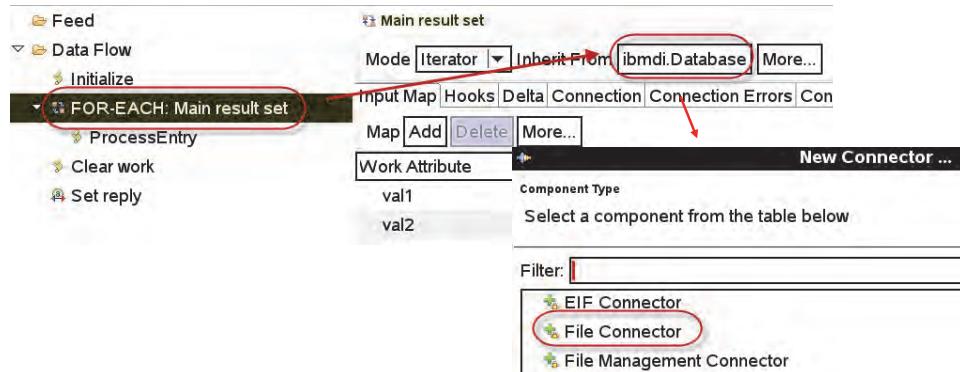


DS_Run_Once assembly line processing flow review

The Run-Once assembly line requires an initialize script. The process entry script is modified so that it finds the record that we are seeking. The variables are rolled up. The work object is cleared, and the selected data is returned.

Modifying the template data connector

- The Feed section does not include a data connector
 - The data connector is included in the FOR-EACH connector loop component
- The template connector loop is configured by default as a Database connector
 - Change the data connector to a file connector: Click the FOR-EACH data flow object, click ibmdi.Database, and select FileConnector



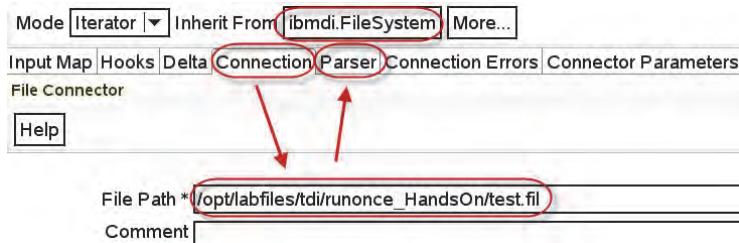
Tivoli Directory Integrator and DASH advanced topics

© Copyright IBM Corporation 2016

Modifying the template data connector

Modifying the file connector

- In this example, the data file is /opt/labfiles/tdi/runonce_HandsOn/test.fil
- Click the Connection tab and enter /opt/tdi/runonce_HandsOn/test.fil in the File Path field
- Click the Parser tab to configure the CSV file parser



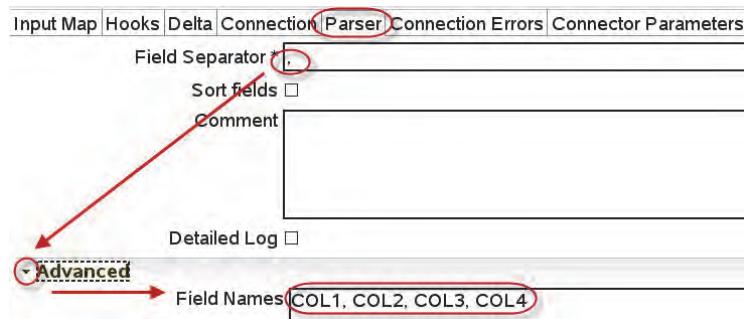
Modifying the file connector

Configure the parser for the File Connector. Set the comma to replace the default semi-colon.

The File Path field can be navigated with clicks until the file is located.

Configuring the CSV file parser

- The data source file is comma-delimited but does not include column header names
 - Enter the comma character (,) in the Field Separator field and click Advanced
 - Enter COL1,COL2,COL3,COL4 in the Field Names field
 - Do not use blank spaces between commas



In the Advanced are, specify the 4 columns for the field names

Testing the file connector

- Click the Input Map tab and click Connect
- Click Next to verify that data is returned in the connector for each column

Work Attribute	Assignment	Schema	
val1	conn.val1	COL1	2
val2	conn.val2	COL2	3
		COL3	4
		COL4	1

Testing the file connector

The initial state has mappings. Map the values the way that you want. In the example, only data values for COL1 and COL2 are of interest.

Mapping input values

- Remove the val1 and val2 attributes that are included with the assembly line template
- Map the COL1 and COL2 attributes and select the Map all Attributes option
 - Only the COL1 and COL2 values are used in this example

The screenshot shows the 'Input Map' configuration window. The 'Assignment' tab is selected. A red box highlights the 'COL1' and 'COL2' entries in the 'Work Attribute' column. The 'Assignment' column shows '(Map all Attributes)' and 'conn.COL1' for 'COL1', and 'conn.COL2' for 'COL2'. The 'Schema' column lists attributes 'COL1', 'COL2', 'COL3', and 'COL4' with small icons next to them.

Work Attribute	Assignment	Schema
*	(Map all Attributes)	Name
COL1	conn.COL1	COL1
COL2	conn.COL2	COL2
		COL3
		COL4

Let's Use a Simple Example

- To complete:
 - Change to CSV parser
 - This CSV file uses a comma as a separator
- The CSV file does not include a column header line
 - Configure the column names in Connection -> Advanced
- The data has 4 columns/attributes
 - Modify the logic to return the attributes of the line found before the line that consists of "0, 0, 0, 0"
- Tips:
 - Convert String to int with val = java.lang.Integer.parseInt(str)
 - To exit a loop: system.exitBranch();
 - To skip to next entry in a loop: system.skipEntry();

Let's Use a Simple Example

Note the tips for how to perform some common functions.

ProcessEntry script component logic review

- The script reads and assigns values for the column1 and column2 values each time a line in the file is processed
 - Assuming the data is valid, first check to see whether both values are zero
 - If the values are not both zero, calculate the number of processed rows
 - Assign the column1 value to lastvar1 and the column2 value to lastvar2
 - When the data=0 condition causes the script to stop processing data from the data source, lastvar1 and lastvar2 contain the target data values

```

if (var1.getValue() == 0 && var2.getValue() == 0) {
    task.logmsg("found the end");
    system.exitBranch();
} else {
    // Sum up some of the data, and increment our row count
    count++;
    task.logmsg("Processing row " + count);
    // Lets also remember values for this record, so that when we hit the "0 0" end
    // we'll have the values from the last row
    lastvar1 = java.lang.Integer.parseInt(var1.getValue());
    task.logmsg("Column1 last value: " + lastvar1);
    lastvar2 = java.lang.Integer.parseInt(var2.getValue());
    task.logmsg("Column2 last value: " + lastvar2);
}

```

ProcessEntry script component logic review

```

// Read in attributes from the current row in the CSV file

var1 = work.getAttribute("COL1");

var2 = work.getAttribute("COL2");

// If we have data

if (typeof(var1) != 'undefined' && typeof(var2) != 'undefined') {

    // If we see a value of 0 and 0 we will consider this a marker to quit processing

    if (var1.getValue() == 0 && var2.getValue() == 0)           task.logmsg("found the end");

        system.exitBranch();

} else {

    // Increment the row count

    count++;

    task.logmsg("Processing row " + count);

    // Save the values for this record, so that when we hit the "0,0,0,0" marker

    // we'll have the values from the row that precedes 0,0,0,0

    lastvar1 = java.lang.Integer.parseInt(var1.getValue());

```

```
task.logmsg("Column1 last value: " + lastvar1);

lastvar2 = java.lang.Integer.parseInt(var2.getValue());

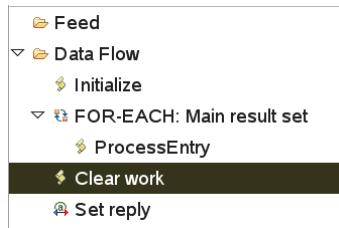
task.logmsg("Column2 last value: " + lastvar2);

}

}
```

Clear work script review

- This script clears the current values in the work object
- Because of the ProcessEntry script logic, the work object values are all zero
- Assign the proper output values in the Set reply component



Configuring the Set reply component

- Map the returned work attributes
- Assign the variables from the process entry script to the work attribute map
 - Values must be assigned from the variables because the work object is empty

Work Attribute	Assignment
*	(Map all Attributes)
COL1	lastvar1
COL2	lastvar2
count	count

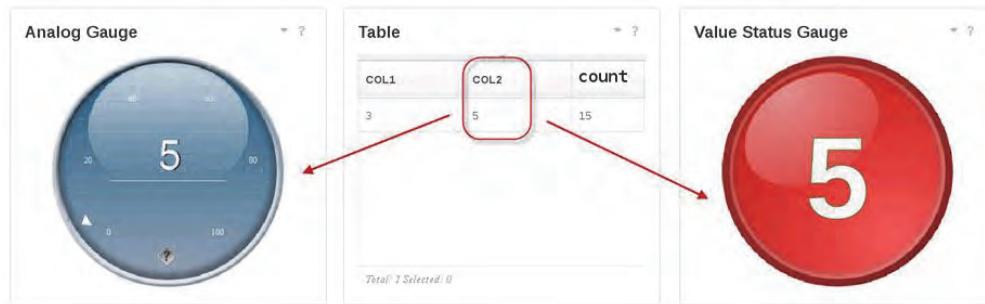
Configuring the set reply component

The values in the set reply component are initially what was brought over from the example. The work attribute values are set to alter what was initially brought over.

Run Once can return more than one previous row of data, depending upon how the script is coded.

Testing the run-once assembly line

- In this example, the assembly line was used with a table, analog gauge, and status value gauge in a dashboard page
 - The COL2 value is used with the analog gauge and value status gauge
 - A threshold value is set in the value status gauge properties



The two gauges are configured with the data.

Exercise 4

Creating a run-once assembly line to generate metric data

Complete Unit 3 Exercise 4 in the exercises guide.

Unit Summary

- Now that you have completed this chapter, you can perform the following tasks:
 - Create Tivoli Directory Integrator UI data providers for various data sources
 - Create helper assembly lines to provide parameter selections within data sets
 - Create helper assembly lines to provide status information from data source evaluation
 - Create helper assembly lines to provide right-click task menus for supported DASH widgets
 - Use the cache connector to use offline data for demonstrations
 - Create a run-once assembly line to evaluate data source data and provide a key performance indicator or status information

Unit 4 DASH and Netcool/Impact integration

IBM Training



Unit 4 DASH and Netcool/Impact integration

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit reviews how to use Impact to transform data from a variety of back-end data sources and then provide the enhanced data to the DASH server as data provider data sets.

Objectives

When you complete this chapter, you can perform the following tasks:

- Configure Netcool/Impact data types and policies as UI data providers
- Configure dashboard widgets with data from a Netcool/Impact data type
- Configure a dashboard widget with data from a Netcool/Impact policy
- Build a dashboard widget by using data from a Netcool/Impact policy variables
- Demonstrate how to use Netcool/Impact output variables to expose data as a UI data provider

Lesson 1 Netcool/Impact overview

IBM Training

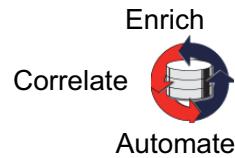


Lesson 1 Netcool/Impact overview

Netcool/Impact overview

Netcool/Impact comprises a set of tools that provide event management and data integration services:

- Event Management
 - Enrichment, suppression, transformation
 - Event Isolation and Correlation
- Automation
 - Creation of trouble tickets, email, problem response
 - Local and remote commands
- Operational Support
 - Create DASH UI data providers to visualize operational data
 - Service Level Agreement (SLA) reports
 - Maintenance mode window



Netcool/Impact overview

Impact solutions, at their core, involve two functions: catching "events" and reacting to those events in some pre-programmed manner.

Events - an event in Impact is some kind of a change in the IT environment that can be captured electronically. Examples of events include:

- a new fault from a DB2 application appearing in Netcool/OMNibus
- the discovery of a new server by a network discovery tool
- a new trouble ticket being opened

Event sources - one of Impact's core features is the ability to capture events from a wide variety of applications and data sources. Impact is most commonly used to capture events in Netcool/OMNibus, but Impact can capture events from the following sources:

- popular relational databases and the applications that use them
- Java Messaging Service (JMS) buses
- Web Services
- Email
- Instant Messaging
- Command line
- IBM Tivoli Network Manager IP Edition

Netcool/Impact has a set of runnable server components that work together to provide event management and integration functionality.

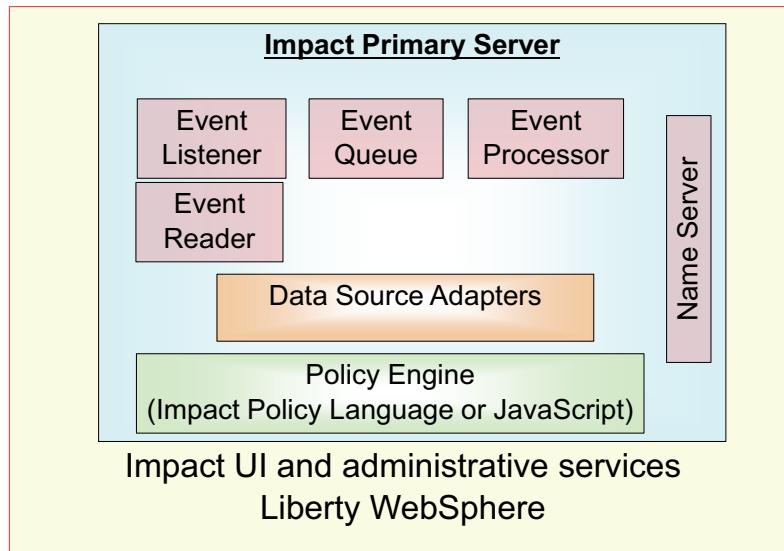
Software system - From a software perspective, you can best understand Netcool/Impact as a set of interrelated, runnable server components, each of which must be installed and configured separately.

Development tool - From an implementation perspective, you can understand Netcool/Impact as a development tool that you use to customize, enhance, and expand the functionality of an existing Netcool installation.

Automation - is the act of setting up a task so that it is performed automatically at certain times or when certain conditions are met.

Uses - You can use Netcool/Impact in a wide variety of ways, depending on the needs of your environment.

High level architecture



High level architecture

The Impact Server overview:

- The Impact Server is responsible for managing the data model, running services, and policies that make up your Netcool/Impact implementation. The Impact Server runs the policies in real time, in response to events that occur in your environment.
- The Impact Server runs as an application inside the WebSphere® Application Server Liberty Core. The Liberty Core is installed and configured automatically as part of the Netcool/Impact installation.
- Important: It is possible to install multiple Impact Servers per system for testing and validation purposes. In a production environment, it is recommended to install one Impact Server per system to achieve better resiliency and performance. Running multiple Impact Servers in a single \$IMPACT_PROFILE, in the application server, is not supported.
- You can run Impact Servers in a clustered server configuration. Such a configuration consists of multiple instances of the Impact Server installed on separate systems but configured in such a way that they act as a single server. This type of configurations provides failover, fallback, replication, and load balancing functions.
- A subcomponent of the Impact Server, the Name Server, provides application registry functions for the Netcool/Impact components. You can use another subcomponent, the JRExec Server, to run external commands, scripts, and applications from within a policy.
- The Impact Server uses a centralized logging feature that incorporates Apache Log4j logging functions.

The Name Server overview:

- The Name Server is a subcomponent of the Impact Server that provides registration functionality for the GUI Server and the Impact Server.
- Applications that use the GUI Server register with the Name Server at startup. The GUI Server uses the information stored in the Name Server when brokering HTTP requests between users' web browsers and the applications. The Name Server also plays a central role in server clustering.

The GUI Server overview

- The GUI Server is responsible for generating the dynamic web-based user interface of Netcool/Impact.
- It brokers requests between users' web browsers and Netcool/Impact. Then, it returns the graphical views that you use to work with the data model, services, and policies.
- The WebSphere® Application Server Liberty Core is installed and configured during the installation if you chose to install the GUI Server as one of the deployment components. The installer sets all the default configuration properties for the server. After the installation, you can change the configuration of the GUI Server by editing its properties files.

"Capturing" events:

- Event Listener > Event Queue > Event Processor.
- Event Reader > Event Queue > Event Processor.
- Impact "captures" an event by loading the event details into memory. The event details depend on the source of the event. In the case of Netcool/OMNibus events, the event details are the alerts.status table's fields that make up the event.
- The components within Impact that capture events are called Readers and Listeners. The key difference between Readers and Listeners is that Readers actively poll a data source for events (changes) while Listeners passively wait for events to be sent to them from the data source. An example Reader is the Netcool/OMNibus Reader which polls OMNibus for changes to the OMNibus database. An example Listener is the Java Messaging Service (JMS) Listener, which simply listens on a JMS topic or queue for new messages.

Reacting to events

- Impact reacts to captured events in its Policy Engine. The events are evaluated against filters and then directed to one or more Impact Policies which define what Impact does with the event. Impact Policies can direct Impact to do things like:
 - correlate event data with data in external sources
 - perform if-then analysis with the event data and data drawn from external sources
 - enrich the original event with external data or with analysis results
 - gateway data to external systems
 - perform escalations and notifications
 - attempt to resolve IT faults automatically

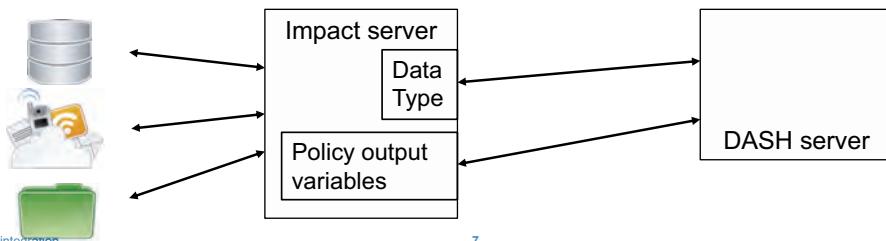
Impact Policies are defined by manually writing Impact Policy scripts (Policies) or using Impact Policy Wizards to generate Impact Policies.

Primary features

- Liberty WebSphere lightweight UI server
 - Supports the administration UI
- DASH server integration
 - Administrative interface federation
 - Operational support through operator view DASH widget
 - UI data provider
- Impact server
 - Data analysis
 - Server cluster support
 - Load balancing and fail over
- Policy Engine
 - JavaScript library and Impact Policy Language (IPL) support
- Event Reader / Listener (Primary server)
- Event Processor
- Multiple Data Source Adapters (DSA)
- Nameserver
 - Registry functionality
 - Identifies the primary and information about the primary server
- Source control
 - Policies and configuration data
 - Internally uses SVN, supports external CVS repositories

Netcool/Impact UI data provider tools

- Impact reads and enhances data from multiple data sources
 - Examples: Databases, flat files, web services, REST
- Impact data processing terminology:
 - Data model: A model of a business that includes data sources, data types, data items, links, and event sources
 - Data type: A data set that is read from a data source query
 - Policies: Data manipulation scripts, written in IPL or JavaScript
- Impact UI data providers expose DASH data with:
 - Data types
 - Policy output variables



Client Use Cases

- Customer 1
 - Need a method to visualize the archived omnibus event data
- Solution:
 - Create an Impact policy which gets omnibus event information from AEL
 - Look up related events from TDW
 - Visualize in the UI as set of tables
- Customer 2
 - Need a method to visualize Windows only metrics from multiple TEMS in a single view
- Solution :
 - Create UI Provider DSA for 2 TEMS Servers
 - Get data from both the TEMS Servers
 - Union the data and display it

These are examples of how a customer might use Netcool/Impact.

Dashboard visualization

- Visualize data from data types and policy output variables
- Large table model support
- Create configuration UI from policy input parameters (requires additional component installation)
- Support for all DASH widgets, including:
 - Tree Widget
 - Topology

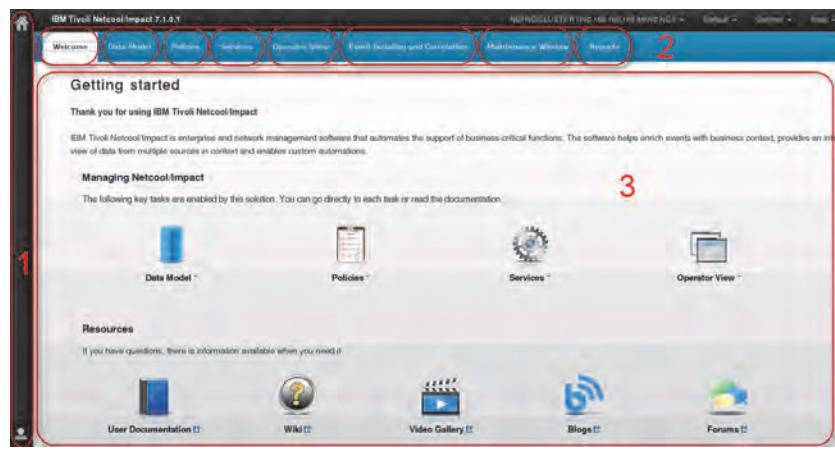


Dashboard visualization

Large table model support is enabled by default for many data sources.

The Netcool/Impact console

- Taskbar: Current user and logout
- Page tabs: Pages are grouped by administrative function
- Current page workspace: Changes based on the selected page and function



The Netcool/Impact console

This is an example of looking at the Netcool/Impact console.

As you select a page tab, it turns white.

Lesson 2 Creating data sets from Netcool/Impact data types

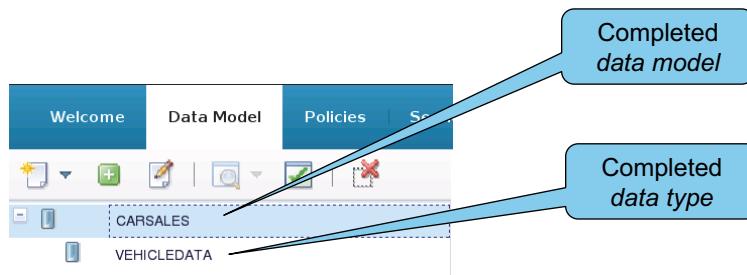
IBM Training

IBM

Lesson 2 Creating data sets from Netcool/Impact data types

Creating an Impact data type

- Complete the following tasks to create an Impact data type:
 - Create a data model object (define a data source)
 - Example: DB2 database CARSALES, host name, authorized user name and password, port number
 - Define one or more data types in the data model (define a data source query)
 - Example: The VEHICLEDATA table



Creating an Impact data type

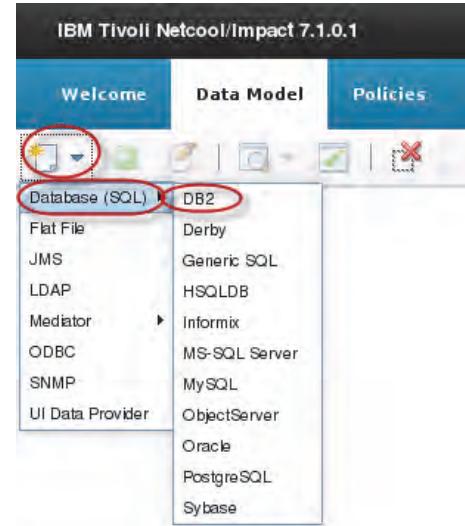
A data model is a model of the business data and metadata that is used in an Netcool/Impact solution.

Creating the data model defines the data source.

The data type defines a data source query.

Creating a data model

- Click the Data Model tab and select the target data source type
- In this example, select a DB2 database type



DASH and Netcool/Impact Integration

13

© Copyright IBM Corporation 2016

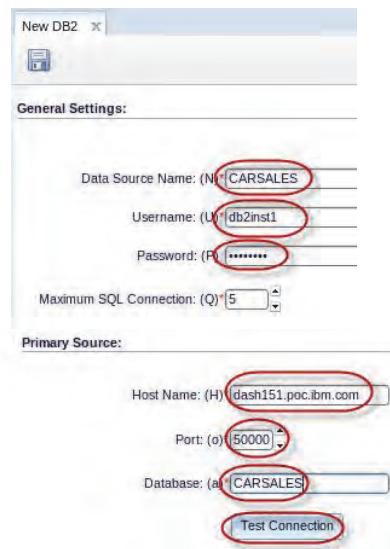
Creating a data model

Note all of the data model types that can be selected.

When DB2 was selected, not the various database types that are displayed.

Configuring the data model data source

- The available parameters vary based on the selected target type
- In this DB2 example:
 - Data Source Name: CARSALES
 - Username: db2inst1
 - Password: object00
 - Maximum SQL Connection: 5
 - Host Name: dash151.poc.ibm.com
 - Port: 50000
 - Database: CARSALES
- You can also define a backup data source



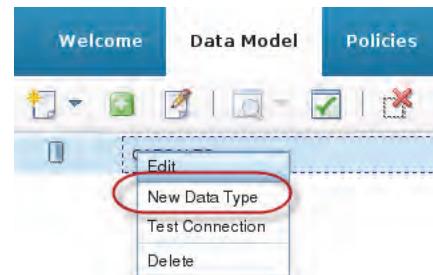
Configuring the data model data source

When creating the New DB2 data model source, the General Settings must be updated. Note the values bounded by a red oval in the slide.

The Test Connection can assist to verify that the database can be accessed with these parameters that were provided.

Creating a data type

- Right-click the data model and select New Data Type
- Assign a name to the data type
 - This value is used as the data set name in a DASH widget quick-edit form
- Select the Access the data through the UI data provider option
- Select a table from the data model



A screenshot of the 'General Settings' configuration page for a new data type. The 'Data Type Name' is set to 'SALESDATA' and the 'Data Source Name' is set to 'CARSALES'. Both settings are circled in red. The 'State' section shows a checked checkbox labeled 'Enabled', which is also circled in red. A callout bubble points to the 'Enabled' checkbox with the text 'Must be selected to be visible to DASH'.

DASH and Netcool/Impact Integration

15

© Copyright IBM Corporation 2016

Creating a data type

Note that the state is checked to be enabled. It must be selected to see this data in DASH.

When you create a data type, you must consider the following settings: For all data types, except the internal data type, you must also select the Key Field check box for the key field. The key field identifies the uniqueness of the data that is displayed by the widget in the console.

You must enable the data type so that it can send data to the UI data provider. To ensure that the data type can be accessed by the UI data provider, open the data type editor and select the Access the data through UI data provider: Enabled check box. After the data model refreshes, the data type is available as a data provider source. The default refresh rate is 5 minutes. However, this setting can be changed. For more information about how to change the refresh rate, see UI data provider customization.

You must select a display name that does not contain any special characters or spaces. To select the display name, click Data Model to open the Data Model tab. Expand the data source that the data type belongs to and click the data type that you want to use. Select the field that you want to use as the display name from the Display Name Field list.

If the data type key field value contains quotation marks ("), the console cannot support the widget that is based on the data type. This means that you cannot click the row that is based on the key field, use it to send an event to another widget or to provide hover preview information. You must use a key field that does not contain quotation marks.

Internal data types

If you use Internal data types, the UI data provider uses the first field in the Additional Fields configuration as the item identifier. You must choose a unique value for the key field, otherwise the UI data provider overwrites your chosen key with the most recent value.

SNMP data types

If you use SNMP data types, you must define a value in the Field Name field in the data type editor. The UI data provider uses the value from the Field Name field as the item identifier. If more than one entry for the same value in Field Name field exists, the UI data provider uses the entry that was created most recently. If you want the UI data provider to use an item identifier that is unique, enter a unique value in the Field Name field for the data type.

Configuring a key field for the data type

- Click Refresh in the Table Description section to see the data type schema
- Double-click the corresponding Key Field cell to select
- You must select at least one key field
 - In this example, WEEKDATE is the data type key field

The screenshot shows the 'Table Description' dialog box. At the top, 'Schemas:' dropdown is set to 'DB2INST1' and 'Tables:' dropdown is set to 'VEHICLEDATA'. Below these are buttons for 'Refresh Fields' (circled in red) and 'Show Deleted Fields'. A red arrow points from the 'Refresh Fields' button to the 'Key Field' column in the table below. The table has columns: ID, Field Name, Format, Display Name, Description, and Key Field. The 'WEEKDATE' row has 'WEEKDATE' in all columns except 'Key Field', where 'Yes' is checked and circled in red.

ID	Field Name	Format	Display Name	Description	Key Field
	WEEKDATE	DATE	WEEKDATE	WEEKDATE	Yes
	VANS	INTEGER	VANS	VANS	No
	SEDANS	INTEGER	SEDANS	SEDANS	No
	PICKUPS	INTEGER	PICKUPS	PICKUPS	No

Configuring a key field for the data type

For all data types, except the internal data type, you must also select the Key Field check box for the key field. The key field identifies the uniqueness of the data that is displayed by the widget in the console. You must choose a key field that makes the record unique.

The WEEKDATE in the example of the CARSALES database is used as the key field because each record has a unique WEEKDATE.

Using an Impact data type UI data provider in DASH

- Search for the name of the data type in the quick-edit search field
- Configure the widget as in previous examples

Column Chart

Select a Dataset

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

(salesdata) Cancel

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

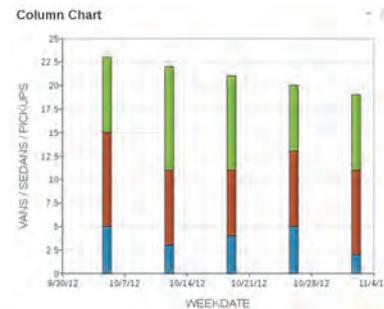
Provider: Impact. TBSMCLUSTER > Datasource: CARSALES

SALES DATA

Datatype from Datasource: CARSALES, name of the datatype: SALES DATA, with tab DB2INST1.VEHICLEDATA.

X axis field: WEEKDATE

Y axis value field: PICKUPS, VANS, SEDANS



Using an Impact data type UI data provider in DASH

Salesdata is entered, resulting in a display of the CARSALES datasource and the SALES DATA dataset. The chart fields are configured.

The results in the column chart look like what was created in a previous unit with Tivoli Directory Integrator using the DB2 database and the CSV file.

Lesson 3 Extending data set content with data type custom output parameters

IBM Training



Lesson 3 Extending data set content with data type custom output parameters

Adding status data to a data type UI data provider

- You can add custom UI data provider output parameters
 - Example: Calculate and add status information on the car sales database

The screenshot shows a user interface titled "Daily Sales Status". At the top right is a search bar with a magnifying glass icon. Below the title, there is a table with five rows. Each row contains a small green square with a white checkmark or an orange square with a red X, followed by a date. The dates listed are 2012-10-05, 2012-10-12, 2012-10-19, 2012-10-26, and 2012-11-02.

	Date
✓	2012-10-05
✓	2012-10-12
✓	2012-10-19
✓	2012-10-26
✗	2012-11-02

Adding status data to a data type UI data provider

This Daily Sales Status is the end result of what you can create using status information as determined from a script.

Creating custom output parameters in the data type

- Edit the data type and add the output parameters in the UI Data Provider section of the table description form
- Example: Add JavaScript (not IPL) code that calculates status information and assigns a status value with the ImpactUICustomValues.put method.
 - if (VANS < 3 && SEDANS < 10 && PICKUPS < 10) {
 - ImpactUICustomValues.put("SALES, Status", "Critical");
 - else {
 - ImpactUICustomValues.put("SALES, Status", "Normal");
- Click the icon to test the syntax and see sample output results

Define Custom Types and Values (JavaScript):

```
if (VANS < 3 && SEDANS < 10 && PICKUPS < 10) {  
ImpactUICustomValues.put("SALES, Status", "Critical"); }  
else {  
ImpactUICustomValues.put("SALES, Status", "Normal"); }
```

Check Syntax and Preview Script Sample Result 

Creating custom output parameters in the data type

Detailed information about the syntax and the use of the ImpactUICustomValues.put function are found in Chapter 13, “Working with the Netcool/Impact UI data provider in the Netcool/Impact Version 7.1.0.1 Solutions Guide.

Click the check syntax to preview the script execution.

Custom data type output parameter results

- In this example, the ImpactUICustomValues.put function added a column called SALES and calculated the status for each WEEKDATE value in the sales database
 - The status values are stored in the SALES column, correlated by the data type key field (WEEKDATE)

UI Data Provider Script Sample Result for Table: SALES DATA

VANS	WEEKDATE	SEDANS	PICKUPS	SALES
5	2012-10-05	10	8	Normal
3	2012-10-12	8	11	Normal
4	2012-10-19	7	10	Normal
5	2012-10-26	8	7	Normal
2	2012-11-02	9	8	Critical

Close

Custom data type output parameter results

The Sales statuses are applied in the SALES column in the table example.

Example: Using custom output parameters with a list widget

- In this example, use the updated VEHICLEDATA UI data provider as the data set for a list widget
 - Select WEEKDATE for the Label value and SALES for the Status value

The screenshot shows the configuration of a list widget. At the top, it displays the selected dataset: Impact_NCICLUSTER > CARSALES > VEHICLEDATA Impact_NCICLUSTER. Below this, it specifies the data format as a table and notes that the dataset is large, has no automatic refresh, and is a remote data provider.

Below the dataset selection, there is a section titled "Map Visualization Attributes to Dataset Columns:". This section contains two rows. The first row maps the "Label" attribute to the "WEEKDATE" column in the dataset. The second row maps the "Status" attribute to the "SALES" column. Both the "Label" and "Status" fields are highlighted with red circles.

The list widget is being configured with the SALES status information.

Lesson 4 Creating Netcool/Impact data sets with policy scripts

IBM Training

IBM

Lesson 4 Creating Netcool/Impact data sets with policy scripts

Creating a UI data provider with an Impact policy

- Similar to Data type UI Provider support visualizing data from a policy
- IPL and JavaScript policies
- Special policy configuration is needed
- Output parameter is similar concept as Input parameter
- Policy data supports all Widgets as Datatypes
- Tree and Topology Widgets are also supported
- Require additional variables in the policy

Creating an Impact policy



- Click the Policies tab and select the type of policy to create
 - Impact Policy Language (IPL)
 - JavaScript
- Optionally, use a wizard to automate the creation of specific types of policies

Creating an Impact policy

From the drop-down window, we selected Use Wizard. Note the processing that can be performed with the event. These are the options that you can select and a description of each option:

- Event Enrichment is the process by which Netcool/Impact monitors an event source for new events, looks up information related to them in an external data source and then adds the information to them.
- Event Notification is the process by which Netcool/Impact monitors an event source for new events and then notifies an administrator or users when a certain event or combination of events occurs. Event Notification policies notify you that an event has occurred. Before you can use the Event Notification policy wizard, configure the e-mail sender service.
- Event Relocation policies can be used to send an event from one central ObjectServer to another ObjectServer.
- Event Suppression policies set a flag in an event in response to a database query. This flag can then be used in a filter to prevent the event from appearing in the Event List.
- X in Y represents X events in Y times. It is the process in which Netcool/Impact monitors an event source for groups of events that occur together and takes the appropriate action based on the event information. X Events in Y policies suppress events until a certain number of identifiable events occur within a specified time period. The X in Y policy wizard tracks how many times a single event with a single identifier is inserted or updated during the time window. If this number reaches (N) then it sends an event to Netcool®/OMNIbus indicating that the threshold

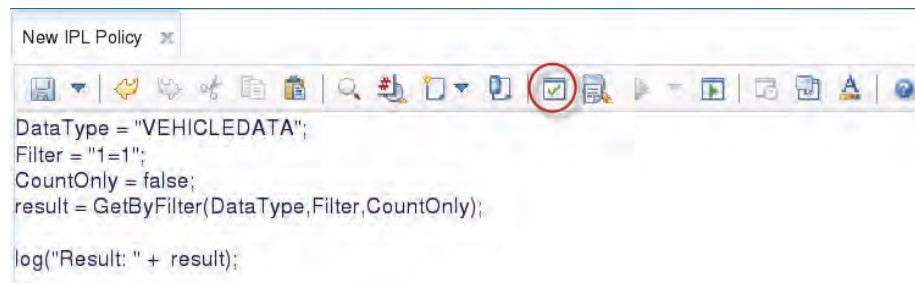
for incidents has been exceeded for the particular event. The XinY policy wizard tracks incidents separately for each of the events that match the filter that triggers the policy.

- You can configure two main parameters in the wizard.
- The number of incidents (N) in an event that will cause a violation.
- The length of the rolling time window in which these (N) incidents must occur to run the violation.
- XML - XML policies are used to read and to extract data from any well-formed XML document.

Web Services - Web Services DSA policies are used to exchange data with external systems, devices, and applications using Web Services interfaces.

Example: Creating a SQL query policy

- In this example, use the GetByFilter function to query the previously created DataType (VEHICLEDATA)
 - The filter value, 1=1, is equivalent to a SELECT * SQL statement
- The data that is returned from the GetByFilter function is stored in the array name result
- Use the Check Syntax tool to verify that the script has no syntax errors and save the policy
 - Assign a policy name when prompted



```
New IPL Policy X
[Toolbar]
[Check Syntax icon circled in red]
[Script Editor]


```

(DataType = "VEHICLEDATA";
Filter = "1=1";
CountOnly = false;
result = GetByFilter(DataType,Filter,CountOnly);

log("Result: " + result);

```


```

Example: Creating a SQL query policy

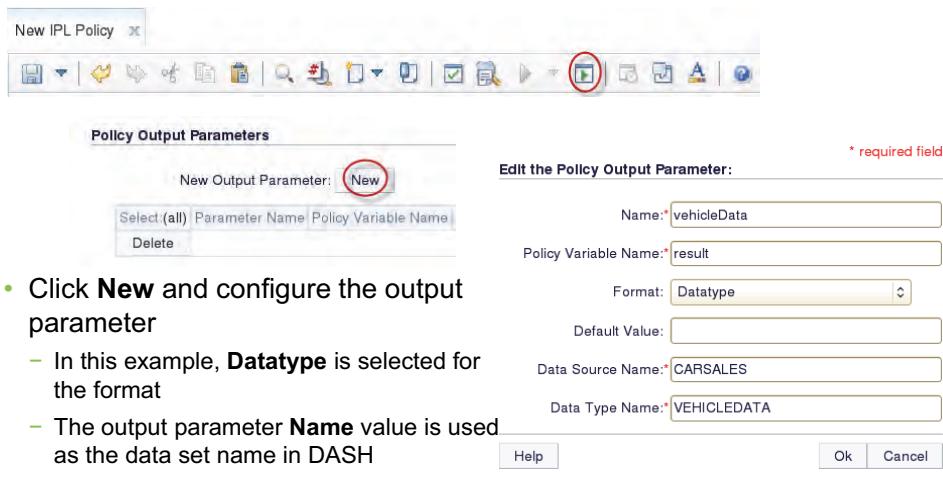
Select everything with the filter value 1=1.

For information about using the GetByFilter function to handle large data sets, open this URL:

http://www-304.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.3/com.ibm.netcoolimpact.doc/common/dita/imdsu_data_provider_large_data_sets.html?lang=en

Creating policy output parameters

- Configure policy output parameters to define the data set that is seen in the DASH widget quick-edit form
- Click the Configure Policy Settings icon in the policy editor toolbar



- Click **New** and configure the output parameter
 - In this example, **Datatype** is selected for the format
 - The output parameter **Name** value is used as the data set name in DASH

DASH and Netcool/Impact Integration

27

© Copyright IBM Corporation 2016

Creating policy output parameters

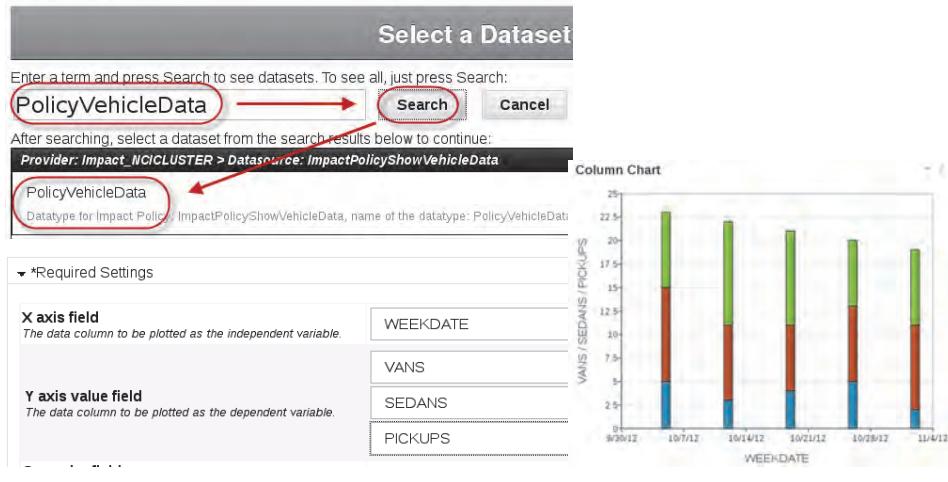
When creating a new output parameter and editing the policy output parameters, these values are used:

- Name - vehicleData
- Policy Variable Name - result
- Format - Datatype
- Data Source Name - CARSALES
- Data type name - VEHICLEDATA

Using an Impact policy UI data provider in DASH

- Search for the name of the data type in the quick-edit search field
- Configure the widget as in previous examples

Column Chart



DASH and Netcool/Impact Integration

28

© Copyright IBM Corporation 2016

Using an Impact policy UI data provider in DASH

This example produces another way to see the same Vehicle data in the stacking of bars in the column chart.

Mandatory Parameter: executePolicy

- Built-in parameter
- Required to execute the policy or not
- If enabled, the policy will be executed whenever the widget is accessed or refreshed.
- If disabled, the widget gets the data from Impact's policy cache.
- Best Practice to control the execution from one Widget only to avoid multiple policy execution

Configure Mandatory Dataset Parameters:

*executePolicy

This parameter indicates whether to execute the Impact policy or not



Mandatory parameter: executePolicy

To run a policy and make the output parameters available to the UI data provider, add executePolicy=true to the following URL:

```
https://<hostname>:<port>/ibm/tivoli/rest/providers/Impact_NCICLUSTER/
datasources/IMPACT_POLICY_<policynname>/datasets/
<policynname>_policy_variables/items?executePolicy=true
```

Lesson 5 Creating data sets from policy script variables

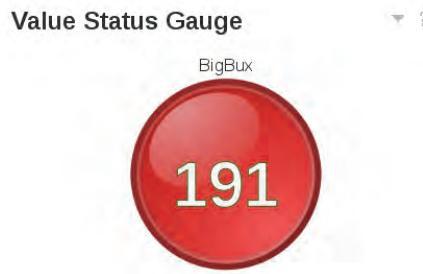
IBM Training



Lesson 5 Creating data sets from policy script variables

Using policy variables as UI data provider output parameters

- In the previous example, the UI data provider output parameter was configured to use data type format
 - All DASH widgets recognize the Impact data type format
- You can also directly use policy variables as output parameter name-value pairs.
- Variable names are shown in DASH with the data set name <Policy_Name>_policy_variables
 - Variable names are shown as the data set column names
- Example: Calculate the number of open troubletickets for a customer and show the value in a value status gauge.



Using policy variables as UI data provider output parameters

The name value pair is <Policy_Name>_policy_variable.

Supported Output Parameter Types

- String
 - Integer
 - Long
 - Float
 - Double
 - Boolean
 - Timestamp
 - Datatype
 - DirectSQL / UI Provider Datatype
 - Impact Object
 - Array Of String
 - Array Of Integer
 - Array Of Long
 - Array Of Float
 - Array Of Double
 - Array Of Boolean
 - Array Of Timestamp
 - Array Of Impact Object

Create a new Policy Output Parameter:

Name:	<input type="text"/>
Policy Variable Name:	<input type="text"/>
Format:	<input type="text" value="String"/> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> String Integer Long Float Double Boolean Timestamp Datatype DirectSQL / UI Provider Datatype Impact Object Array Of String Array Of Integer Array Of Long Array Of Float Array Of Double Array Of Boolean Array Of Timestamp Array Of Impact Object </div>
Default Value:	<input type="text"/>
Data Source Name:	<input type="text"/>
Data Type Name:	<input type="text"/>

Help

Output Parameter notes

- DirectSQL / UI Provider Datatype, Impact Object, Array Of Impact Object
- When selected, a new option is enabled
 - A schema definition is mandatory for these types of output parameters
- Field - Type pairs are required for any field that should be visualized in the widget



- DirectSQL /UI Provider requires a key field
- Array Of Impact Object uses a built-in key UIObjectID

Output parameter notes

This URL has information about Visualizing data from the UI data provider in the console:

http://www-304.ibm.com/support/knowledgecenter/SSSHYH_7.1.0.3/com.ibm.netcoolimpact.doc/solution/uiddataprovider_scen_overview_c.html?lang=en

Creating the trouble ticket data type

- The policy will use data from a trouble ticket database
- Create a data model connection to the database called TroubleTickets
- Create a data type that pulls all of the data from the TICKETS table and call the data type TicketCount
- Use the TicketID column as the key field for the data type

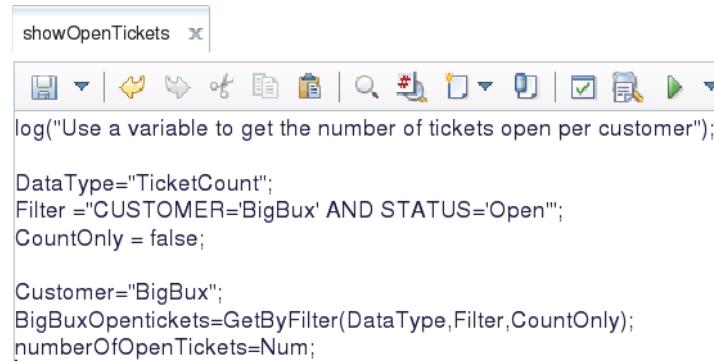
Select:(all)	ID	Field Name	Format	Display Name	Description	Key Field
<input type="checkbox"/>	TICKETID	TICKETID	INTEGER	TICKETID	TICKETID	<input checked="" type="checkbox"/>
<input type="checkbox"/>	HIGHLEVELSERVICEID	HIGHLEVELSERVICEID	STRING	HIGHLEVELSERVICEID	HIGHLEVELSERVICEID	<input type="checkbox"/>
<input type="checkbox"/>	LOWLEVELSERVICEID	LOWLEVELSERVICEID	STRING	LOWLEVELSERVICEID	LOWLEVELSERVICEID	<input type="checkbox"/>
<input type="checkbox"/>	CUSTOMER	CUSTOMER	STRING	CUSTOMER	CUSTOMER	<input type="checkbox"/>
<input type="checkbox"/>	TICKETTYPE	TICKETTYPE	STRING	TICKETTYPE	TICKETTYPE	<input type="checkbox"/>
<input type="checkbox"/>	REGION	REGION	STRING	REGION	REGION	<input type="checkbox"/>
<input type="checkbox"/>	SEVERITY	SEVERITY	INTEGER	SEVERITY	SEVERITY	<input type="checkbox"/>
<input type="checkbox"/>	STATUS	STATUS	STRING	STATUS	STATUS	<input type="checkbox"/>
<input type="checkbox"/>	DATEOPENED	DATEOPENED	INTEGER	DATEOPENED	DATEOPENED	<input type="checkbox"/>
<input type="checkbox"/>	SUMMARYTEXT	SUMMARYTEXT	STRING	SUMMARYTEXT	SUMMARYTEXT	<input type="checkbox"/>
...						

Creating the trouble ticket data type

TICKETID is the unique field to use as the key field.

Creating the policy

- You need to calculate all of the open tickets for the customer, BigBux bank
- Use the getByFilter command with a filter that searches for all records where CUSTOMER='BigBux' and STATUS='Open'
 - Num is an internal variable that counts the records returned by the getByFilter command
 - Expose the variables Customer and numberOfOpenTickets as output parameters



```
showOpenTickets x
log("Use a variable to get the number of tickets open per customer");

DataType="TicketCount";
Filter ="CUSTOMER='BigBux' AND STATUS='Open'";
CountOnly = false;

Customer="BigBux";
BigBuxOpentickets=GetByFilter(DataType,Filter,CountOnly);
numberOfOpenTickets=Num;
```

DASH and Netcool/Impact Integration

35

© Copyright IBM Corporation 2016

Creating the policy

This script starts with logging a message. The log function is very useful when debugging a script.

Creating the output parameters

- Configure two output parameters, one called CustomerOpenTickets (a string value) and NumOpenTickets (an integer value)
 - You do not need to configure a data source or data type for these variable output parameters

The screenshot shows the 'Edit the Policy Output Parameter' screen for two parameters. On the left, 'CustomerOpenTickets' is defined as a String format, while 'NumOpenTickets' is defined as an Integer format. Both parameters have their Policy Variable Name set to 'Customer' and 'numberOpenTickets' respectively. Below this, the 'Policy Output Parameters' section lists both parameters with their details: CustomerOpenTickets (Customer, String) and NumOpenTickets (numberOpenTickets, Integer). A 'New' button is available to add more parameters.

Select:(all)	Parameter Name	Policy Variable Name	Format	Default Value	Data Source Name	Data Type Name
<input type="checkbox"/>	CustomerOpenTickets	Customer	String			
<input type="checkbox"/>	NumOpenTickets	numberOpenTickets	Integer			

DASH and Netcool/Impact Integration

36

© Copyright IBM Corporation 2016

Creating the output parameters

The two output parameters are CustomerOpenTickets and NumOpenTickets. The formats of these two parameters are different, One parameter is a string or text value and the other parameter is a numeric integer value.

Using a variable policy data set in a DASH widget

- Search for tickets and select showOpenTickets_policy_variables
 - Use NumOpenTickets for the gauge value and use CustomerOpenTickets for the label above the gauge

The screenshot shows the 'Select a Dataset' dialog at the top, where 'tickets' is searched and the 'showOpenTickets_policy_variables' dataset is selected. Below it, the 'Map Visualization Attributes to Dataset Columns' page is shown, with 'Value' set to 'NumOpenTickets' and 'Label above Gauge' set to 'CustomerOpenTickets'.

Select a Dataset

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

After searching, select a dataset from the search results below to continue:
Provider: Impact_NCICLUSTER > Datasource: showOpenTickets
showOpenTickets_policy_variables

Map Visualization Attributes to Dataset Columns:

Value: NumOpenTickets

Label above Gauge: CustomerOpenTickets

Label at leading edge: None

DASH and Netcool/Impact Integration 37 © Copyright IBM Corporation 2016

Using a variable policy data set in a DASH widget

The showOpenTickets dataset has _policy_variables appended to it.

Exercise 1 **Creating a Netcool/Impact data type data set**

Exercise 2 **Adding status information to a data type UI data provider**

Exercise 3 **Using a Netcool/Impact policy to create a data set**

Exercise 4 **Using policy variables as UI data provider output parameters**

Exercises

Complete Unit 4 exercises 1 through 4 from the exercise guide.

Summary

Now that you have completed this chapter, you can perform the following tasks:

- Configure Netcool/Impact data types and policies as UI data providers
- Configure dashboard widgets with data from a Netcool/Impact data type
- Configure a dashboard widget with data from a Netcool/Impact policy
- Build a dashboard widget by using data from a Netcool/Impact policy variables
- Demonstrate how to use Netcool/Impact output variables to expose data as a UI data provider

Unit 5 Using the Jazz for Service Management web widget with business dashboards

IBM Training



Unit 5 Using the Jazz for Service Management web widget with business dashboards

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

The DASH Web widget is very versatile for serving web-based content in dashboard pages. In this unit, you learn how to use the built-in myBox web application container to host images, jsp files, and html on the DASH server. You learn how to use hotspot events and command substitution to dynamically modify widget data content, based on the selection of hotspots and other widgets.

Objectives

After completing this unit, you should be able to perform the following tasks:

- Describe how to configure and use the Jazz for Service Management web widget in business dashboards
- Use debugging tools to analyze and examine event parameters from dashboard widgets
- Substitute NodeClickedOn and displayURL widget event parameter values in web widget addresses to send context information
- Use web widgets to add non-IBM application consoles to dashboard pages
- Use hotspot widgets to change the data that is shown in web widgets and create interactive dashboard pages
- Use the SmartText widget to dynamically create and serve HTML text and data set data

Lesson 1 Web widget overview

IBM Training

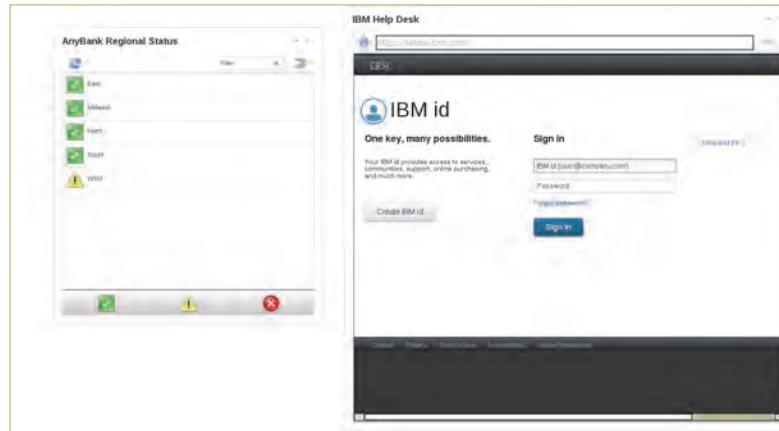


Lesson 1 Web widget overview

Dashboards and web-based content

Use web widgets to:

- Add web-based content to a dashboard page
- Quickly add web-based application consoles to business dashboards
- Enhance and complement dashboard pages with graphic images, HTML, and JSP pages



Using the Jazz for Service Management web widget with business dashboards

4

© Copyright IBM Corporation 2016

Dashboards and web-based content

This web-based content can include locally hosted or remote content.

An web-based content example to add might be stock ticker feeds.

Web widget architecture

- Shows web-addressable (local or remote) content in an inline frame, or iframe, object
- Does not use UI data provider connections
- Inline frames can be embedded in an HTML document
 - Use unique iframe names, or IDs, with multiple web widgets on a page
 - The iframe ID provides an addressable target ID for widget events and JSP scripts



Using the Jazz for Service Management web widget with business dashboards

5

© Copyright IBM Corporation 2016

Web widget architecture

As long as the web-based content that you want to show works okay with an in-line frame, then you can add it to your page.

Company logos and run books with web-based content might be something to add quickly.

Web widget configuration

Web Widget

Widget title:

Enter a Web widget title name

Optional widget title

Home page (use http:// for remote hosts or relative URLs for the dashboard server):

https://www.ibm.com/

Web address for the inline frame

Help page:

To provide a custom help topic, enter a link to a help topic, for example, webwidget.html.

Optional help page web link

HTML iFrame name:

Enter a iFrame name

Optional iframe name

Note: An iFrame name must be unique. Do not use the same iFrame name for multiple widgets.

Choose to show a web address toolbar

Check Show a browser control toolbar to enable a browser navigation toolbar in the Web Widget.

Show a browser control toolbar

Select to show user widget controls

By default, users cannot personalize their Web Widget settings. To allow users to personalize a setting, check the relevant option:

Widget title

Home page (use http:// for remote hosts or relative URLs for the dashboard server)

Help page

Browser control toolbar

Web widget configuration

The highlighted area are configure for a web widget.

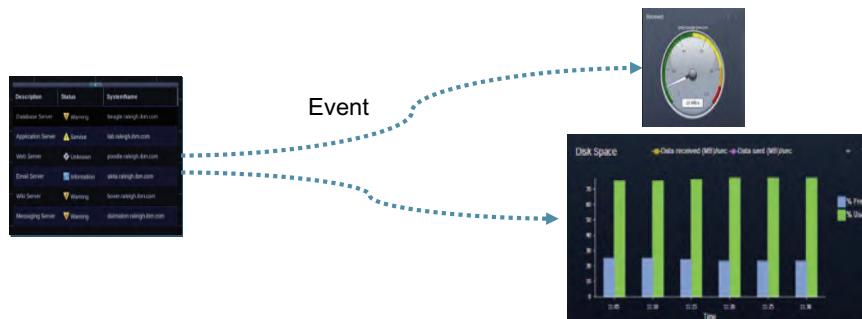
If the web site is remote, it must use https, not http.

You can use relative addressing with MyBox for web-based widget information.

The iFrame name is unique and can be specified in hot spots

Widget events overview

- Widget events are data parameters that are published by selecting widget content
 - Not all widgets publish events
 - Compatible widgets can subscribe to receive events
- Example: A page contains a table, analog gauge, and bar chart widget
Clicking a row in the table widget sends a *NodeClickedOn* event to the analog gauge and chart widgets
The content that is shown in the gauge and chart widgets change, based on the received event



Using the Jazz for Service Management web widget with business dashboards

7

© Copyright IBM Corporation 2016

Widget events overview

Most widgets can publish events. It broadcasts events to other widgets on the page. You can also specify that an event go to a particular target, using an event wire.

Modifying the widget event publish and subscribe configuration

1. Click the **Edit options** icon in the upper right of the widget and select **Events**
2. Enable or disable the published or subscribed events

The available published and subscribed events varies by widget type

The screenshot illustrates the configuration of event publishing and subscribing for two different widget types: 'Web widget' and 'List'.

Web widget Configuration:

- Published Events:** Shows a single entry: "DisplayURL".
- Subscribed Events:** Shows three checked entries: "DisplayURL", "NodeClickedOn", and "dataRefresh".

List Configuration:

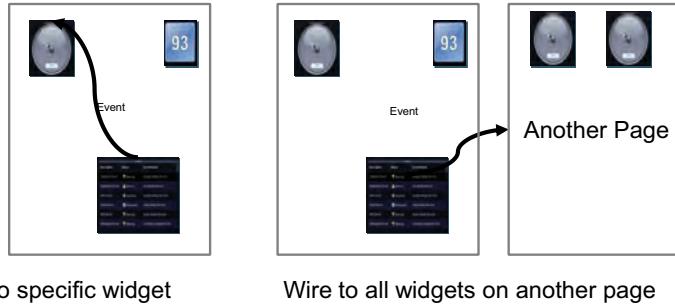
- Published Events:** Shows one checked entry: "NodeClickedOn".
- Subscribed Events:** Shows two entries: "NodeClickedOn" (unchecked) and "dataRefresh" (checked).

Modifying the widget event publish and subscribe configuration

The events option will show how or if the events are handled.

Event wires overview

- Event wires direct events between widgets and pages
- Use event wires to create navigational paths between dashboard pages and widgets
Optionally configure a wire to open and load the target page
- Publish events to another page or a specific widget on a page

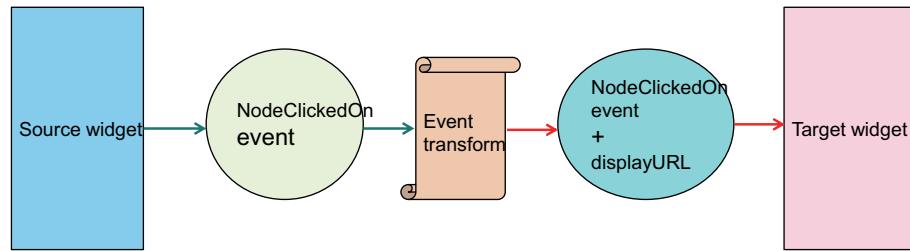


Event wires overview

Event wires are used to specify that a particular event is targeting another page or widget.

Event transformations

- *Event transformation* is a tool to enrich a widget event before the target receives the event
- Used when the event data is not compatible with the target widget
 - Example: A web widget expects a displayURL parameter in a NodeClickedOn event, but the source event does not include the displayURL parameter
- Requires creation of an XML configuration file and corresponding JavaScript file
- Due to complexity and problems with high availability environments, *parameter substitution* can be a simpler alternative method



Using the Jazz for Service Management web widget with business dashboards

10

© Copyright IBM Corporation 2016

Event transformations

Event transformations can alter the event data to enrich an event before the target receives it.

Because these are difficult to maintain, parameter substitution is a better alternative.

DASH tools overview

- A set of advanced dashboard development troubleshooting tools: Event Spy widget, NodeClickedOn widget, and Data Detective
- Designed for use with internet browser debugging tools
- Free download available at the IBM developerWorks Jazz for Service Management UI Services wiki: https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W3d82b8c8fc4b_482e_94c3_34d88b299e0e/page/Dash%20Tools
- Requires community membership

The screenshot displays three widgets side-by-side:

- Event Spy**: A table showing event counts for various actions: NodeClickedOn (1), launchPage (None), portletResize (2), dataRefresh (None), and Total Events (3). It also shows a timestamp of 10:07:26.
- Node Clicked On**: A table listing node click parameters with their values. The columns are Event Field and Value.
- Data Detective**: A REST provider query interface. The URL is /ibm/tivoli/rest/providers. The response shows a JSON object with filteredRows: 13, identifier: "id", items: [{}], and baseUrl: "https://jazz01.tivoli.edu:16311/ibm/tivo".

Using the Jazz for Service Management web widget with business dashboards

11

© Copyright IBM Corporation 2016

DASH tools overview

These DASH tools provide an easy way to track events that have been generated on a page. The tools are automatically included with the product. The link shows where the latest versions of the tools can be downloaded.

The NodeClickOn event widget is a two-column table that is populated with the event field, parameter name, and value.

The Data Detective lets you query a REST provider to ensure that the widget is working.

Adding Event Spy widget to a page

- Edit a page and drag the Event Spy widget to the page
 - Provides a visual summary of event activity
 - Automatically adds event data to browser debugger console
- No widget configuration options

TBSM Status	Display Name	cpuUtilization (Status)	cpuUtilization
✓ Good	■ StockTrader	✓ Good	0
✓ Good	■ StockTrader_London	✓ Good	0
✓ Good	■ StockTrader_Tokyo	✓ Good	0
✓ Good	■ Zurich\$StockTrader	✓ Good	0

Adding Event Spy widget to a page

The Event Spy widget keeps running counts of whatever type of event is generated.

Viewing events in the debugger console

- Use with a browser debugger tool, such as Firebug
 - Event context is added to the console log
 - Search for log entries that start with the text string, ***E* EVENT SPY**
- Click the NodeClickedOn link in the console to view the event data

The screenshot shows two panels from a browser developer tools interface. The top panel is a 'Console' tab showing log entries:

```
>>> EVENT SPY: page kick off
<<< EVENT SPY: page kick off
*E* EVENT SPY(1): portletResize(1) Object { name="http://ibm.com/isclite#portletResize" }
*E* EVENT SPY(2): portletResize(2) Object { name="http://ibm.com/isclite#portletResize" }
! *E* EVENT SPY(3): NodeClickedOn(1) Object { payload=..., name="http://ibm.com/tip#NodeClickedOn" } internal
```

The bottom panel is an 'Object' inspector for the NodeClickedOn event:

window > Object

Internal_TIP_Event_Identifier_Key	-9007199254740988
name	"http://ibm.com/tip#NodeClickedOn"
payload	Object { resources=[0] }
resources	[Object { viewName="RawEvents", forceOverwrite=true, filter=... }, Object { viewName="RawEvents", forceOverwrite=true, filter=... }]
ParentServiceKey	2508
ServiceInstanceIdName	"Asia"
TBSMServiceContextsCollectionSelectedServices	"AsiaABCBank"
TbsmServiceInstanceDisplayName	"Asia"
TbsmServiceInstanceIdName	"AsiaABCBank"
filterName	"RawEvents_2508"
filterType	"user_transient"
forceOverwrite	true
registerFilter	true
sql	"Class != 12000 and RAD_...re Service = '1:2508')
viewName	"RawEvents"
viewType	"system"

Two specific objects are highlighted with red boxes and labeled:

- NodeClickedOn parameters**: Points to the 'resources' array entry.
- NodeClickedOn parameter values**: Points to the 'resources' array entry's value.

Using the Jazz for Service Management web widget with business dashboards

13

© Copyright IBM Corporation 2016

Viewing events in the debugger console

You can use a browser event like Firebug to see the information.

NodeClickedOn widget overview

- The NodeClickedOn widget is a simpler alternative to using the EventSpy-debugger combination
- Add the widget to a page and generate an event
 - The event parameters and values are shown in a table

California Office Sales

WEEKDATE	
Oct 5, 2012 12:00:00	
Oct 12, 2012 12:00:00	Oct 12, 2012 12:00:00
Oct 19, 2012 12:00:00	
Oct 26, 2012 12:00:00	
Nov 2, 2012 12:00:00	

Node Clicked On

Event Field	
LOCATION	CA
TOTAL	11
WEEKDATE	2012-10-12
...	

Using the Jazz for Service Management web widget with business dashboards

14

© Copyright IBM Corporation 2016

NodeClickedOn widget overview

Drag and drop a nodeClickedOn widget to your event page. It won't require configuring. If you generate a NodeClickedOn event, the information will appear in the NodeClickedOn widget table.

Instructor demonstration

Examining widget event parameters



Exercise 1 Examining widget event parameters

Student exercises

Complete Unit 5 Exercise 1 in the exercise guide.

Lesson 2 Web widget event parameter substitution

IBM Training

IBM

Lesson 2 Web widget event parameter substitution

Parameter substitution in the web widget home page

- Substitute NodeClickedOn event parameter values in the web widget home page configuration
- Substitution parameters are enclosed with two per cent sign symbols: %%<ParameterName>%%
- For example:
 - A table widget generates a NodeClickedOn event that includes a **City** parameter with a value of **Chicago**
 - The web widget home page address is configured as **/myBox/%%City%%.html**
 - Click the table widget and the web widget opens the local server file **/myBox/Chicago.html**



Parameter substitution in the web widget home page

You can use NodeClickedOn event parameters in a web page. Bound the parameter name by two percent signs on each side. The event from the NodeClickedOn event can be substituted into the web page name, possibly resulting in a different web page to be loaded.

Parameter Substitution Example

Web widget home page configuration:
 /myBox/html/runbook/%%ORIGINNODE%%.html

The screenshot shows a web browser displaying a table of system monitoring data. A blue arrow points from the table to a callout box containing the text: "Table widget with IBM Tivoli Monitoring agent data". Another blue arrow points from the table to a configuration dialog titled "Enter a Web widget title name". The dialog contains the URL: "/myBox/html/runbook/jazz01:LZ.html". A red box highlights the row in the table where the system name is "jazz01:LZ". A blue box highlights the URL in the configuration dialog.

System Name	Time Stamp	CPU ID	User CPU (%)	User Idle CPU (%)	System CPU (%)	Idle CPU (%)
jazz01:LZ	Jul 10, 2014 12:02:02	Aggregate	7%	0%	1%	92%
jazz01:LZ	Jul 10, 2014 12:02:02	0	5%	0%	1%	94%
jazz01:LZ	Jul 10, 2014 12:02:02	3	5%	0%	1%	94%
jazz01:LZ	Jul 10, 2014 12:02:02	Aggregate	2%	0%	1%	97%
jazz01:LZ	Jul 10, 2014 12:02:02	0	5%	0%	1%	94%
jazz01:LZ	Jul 10, 2014 12:02:02	3	5%	0%	1%	94%

Before clicking table widget row:
 /myBox/html/runbook/.html

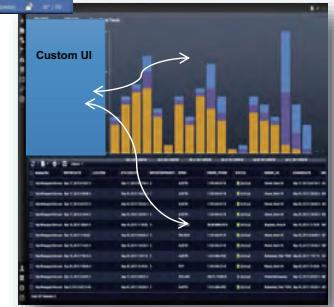
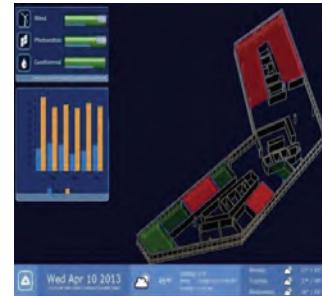
After clicking table widget row:
 /myBox/html/runbook/jazz01:LZ.html

Parameter Substitution Example

Note how “jazz01:LZ”, a managed system name representing a Linux OS agent running on host jazz01, has been substituted into the HTML page name.

Serving web content with myBox web archive file

- A web archive (WAR) application, added in Jazz for Service Management 1.1 Fix Pack 3
- The myBox WAR application is used to locally serve web content, such as:
 - JavaServer Pages (JSPs)
 - Java servlets
 - Static image files
 - JavaScript files
 - Cascading Style Sheet (css) files
 - HTML files
 - Other static content
- You can use myBox content to create custom dashboard elements
- Includes a simple script to manage content
 - Updates do not require a server restart



Serving web content with myBox web archive file

Many other kinds of content can be brought into the web widget. Anything that is addressable with a standard URL can be hosted in the web widget.

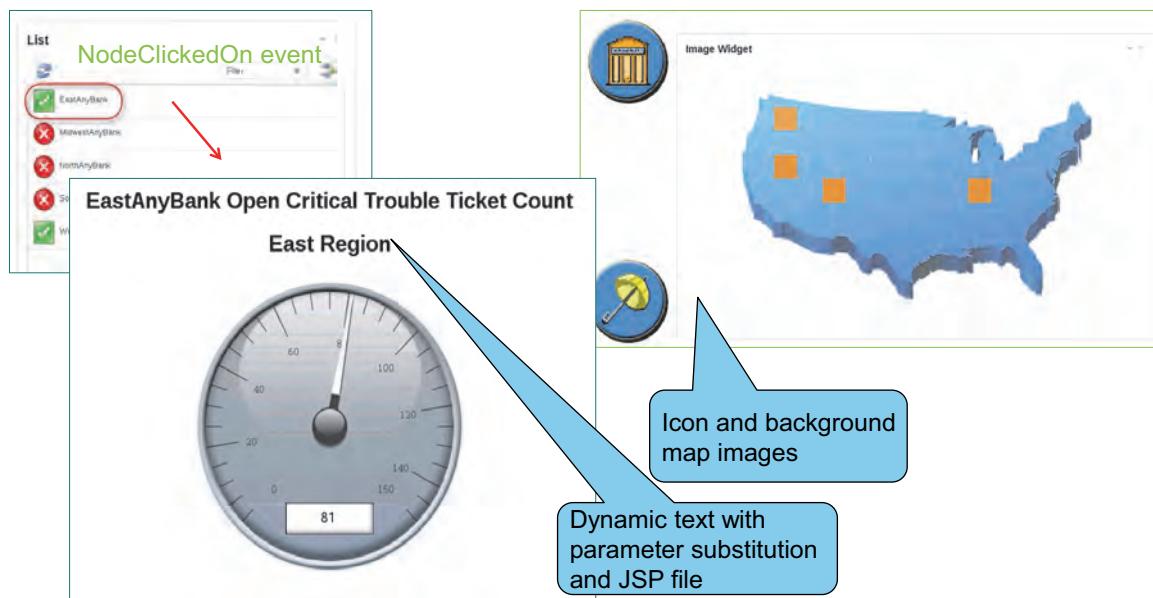
Managing myBox content

- Add files or directories to the **\$JAZZHOME/ui/myBox/web_files** directory
- Add files or directories to **\$JAZZHOME/ui/myBox/web_files/secure** to require an authenticated console connection
- To update the web content, enter the command:
**\$JAZZHOME/ui/myBox/deployMyBox.sh **
**-username <username> **
**-password <password> **
- You must use the command when any myBox file is modified
 - The files are compiled and added to the DASH server jvm

Managing myBox content

This slide shows how to manage the myBox directory.

Examples: Serving files with the myBox web archive



Examples: Serving files with the myBox web archive

These widgets are examples of serving data from the myBox web archive. The example with the gauge uses substitutions of the East bank information. The map substitutes image widget information based on the hot spot values.

Lesson 3 Web widget parameter substitution with JavaServer pages

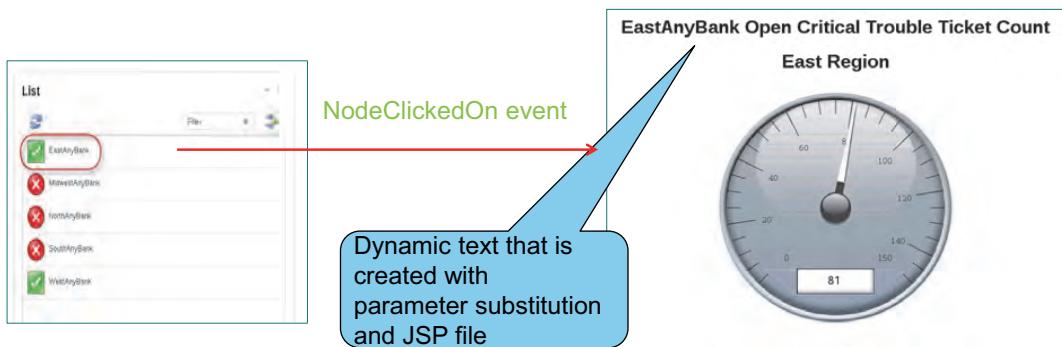
IBM Training

IBM

Lesson 3 Web widget parameter substitution with JavaServer pages

Using web widgets with JSP scripts

- JSP scripts are used to dynamically create web pages
 - Use the myBox web server to host local JSP files
 - You can use widget event parameter substitution to create input parameters to the JSP file
- Example: Use parameter substitution with a JSP file to send contextual data to a page text banner



Using web widgets with JSP scripts

The JSP file generates an HTML. The NodeClickOn event parameter can be used to provide input into the JSP file. The context information generates dynamic text into the gauge widget heading information.

Configuring a web widget to use a JSP file

- The web widget home page configuration points to the JSP file, either with a remote or local URL
 - Script input parameters are included in the home page address, delimited with a question mark (?) character
 - Multiple input parameters are delimited with the ampersand (&) character
- You can use widget event substitution with the parameter values
 - Example: /myBox/filename.jsp?jspParam1=%eventParam1%&jspParam2=%eventParam2%
- Use the request.getParameter() method to use the substituted parameter value in the JSP file
 - Example: A NodeClickedOn event contains two event parameters, eventParam1=EastAnyBank and eventParam2=East. The following table shows the JSP expressions and expression values that are used in the JSP file:

JSP expression	Expression value
<%=request.getParameter("jspParam1")%>	EastAnyBank
<%=request.getParameter("jspParam2")%>	East

Configuring a web widget to use a JSP file

Everything after the question mark “?” in the event substitution are the parameters to substitute. These substitutions are used in the JSP file values.

Exercise 2 **Using parameter substitution with** **web widgets**

Exercise 3 **Using parameter substitution to** **create and show dynamic text**

Exercises

Complete Unit 4 Exercises 2 and 3 from the exercises guide.

Lesson 4 Using web widgets to show non-IBM application consoles

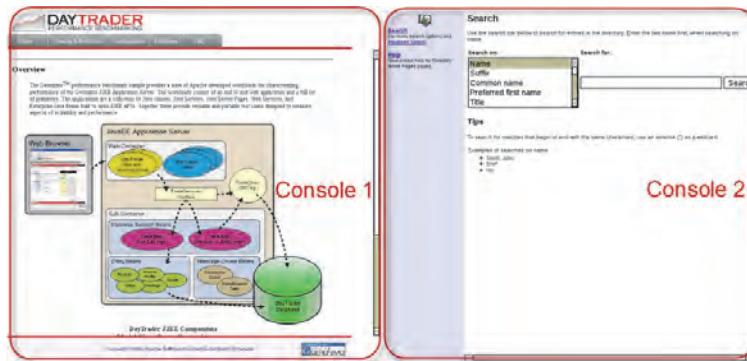
IBM Training



Lesson 4 Using web widgets to show non-IBM application consoles

Adding application consoles to dashboard pages

- Use a web widget to add a web-based customer application user interface to a dashboard page
- Benefits:
 - Does not require changing application console code
 - The web widget console can be linked to other web pages with widget events
 - Provides time to transition to widget-based dashboard replacement



Using the Jazz for Service Management web widget with business dashboards

28

© Copyright IBM Corporation 2016

Adding application consoles to dashboard pages

As long as a console is accessible through a browser, you can add that console to the web widget. A legacy console can be launched or linked to an existing customer application with the web widget.

Web widget configuration with application consoles

- Configure the web widget home page address with the application console web address
- The application console must be web-based and compatible with iframes
- If the console does not support SSO, the user is challenged with a second logon request



Web widget configuration with application consoles

Some applications might not behave properly with an iFrame and should not be used.

Lesson 5 Using web widgets to show non-IBM application consoles

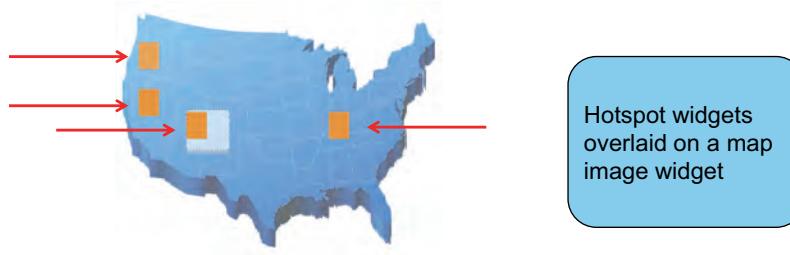
IBM Training



Lesson 5 Using web widgets as a hotspot widget target

Hotspot widget overview

- Use hotspot widgets to:
 - Supplement dashboard page information with cursor tooltips or pop-up messages
 - Provide navigation points to portal or web pages
 - Generate NodeClickedOn or displayURL events
- Hotspot widgets are configured as a square or rectangle or with a static graphic image
- Hotspot widgets can be overlaid on other widgets

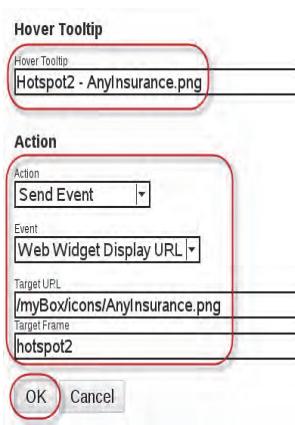


Hotspot widget overview

This is similar to using the hot spots in previous exercises. If a web widget receives a displayURL event, it will use that event as the URL. The map on the page shows four hot spots.

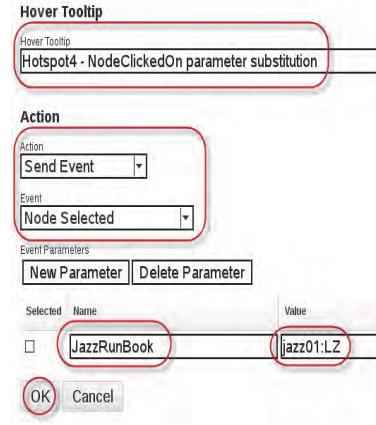
Configuring hotspot widgets to send events

- Configure target web widgets with unique iframe IDs
- Configure the hotspot widget to send displayURL or NodeClickedOn events to one or more iframe IDs



Example: displayURL configuration

Using the Jazz for Service Management web widget with business dashboards



Example: NodeClickedOn configuration

© Copyright IBM Corporation 2016

Configuring hotspot widgets to send events

These are two examples of hot spots that can be generated. The left side shows Hotspot2 - AnyInsurance.png. It uses the Web Widget Display URL event and the Target frame.

The right side example shows the Node Selected event, and generates the parameter JazzRunBook with the value Jazz01:LZ.

Lesson 6 Using web widgets as a hotspot widget target

IBM Training

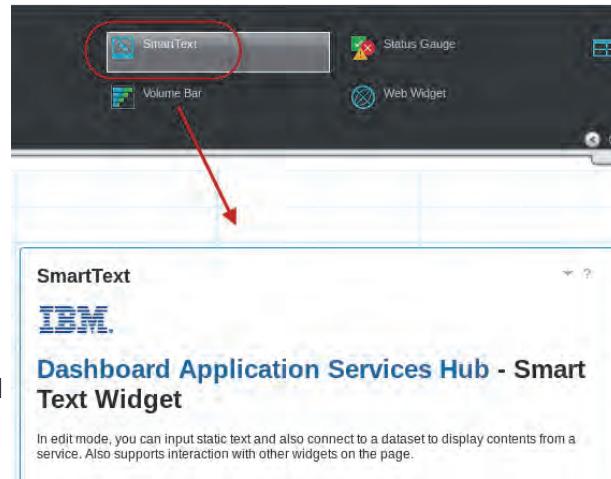
IBM

Lesson 6 Using the SmartText widget to create dynamic text

The SmartText widget can create dynamic text.

Smart text widget overview

- The widget combines an HTML text editor with dynamic links to a selected data set
- A data set target can be selected in the widget configuration or supplied by context events
- Includes a table insert function
- Switch to HTML source view to provide advanced HTML formatting tags



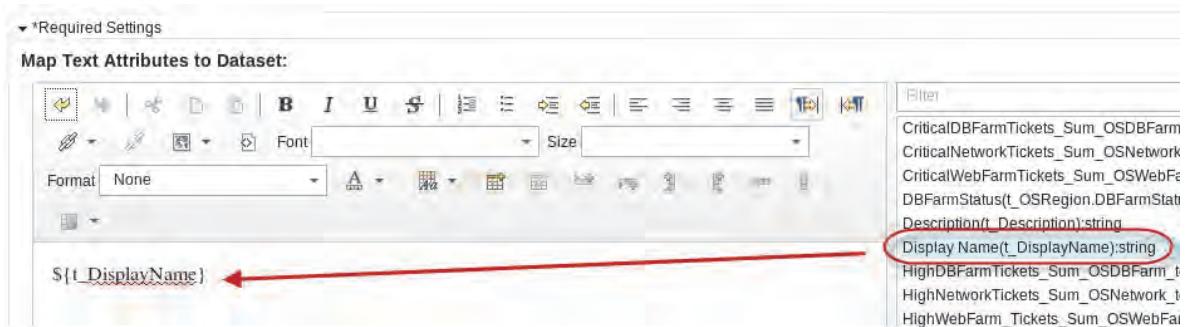
Smart text widget overview

The Smart text widget has an HTML editor that can be used with dynamic links.

Table widgets can consume a lot of space on the page. The SmartText widget is an alternative that can consume less space.

Inserting data set tags

- Select a data set attribute to insert a variable representation into the text editor
- Variable values are dynamically replaced when the widget is rendered, either with a NodeClickedOn event value or from the widget configuration
- The data set attribute list includes the attribute data type

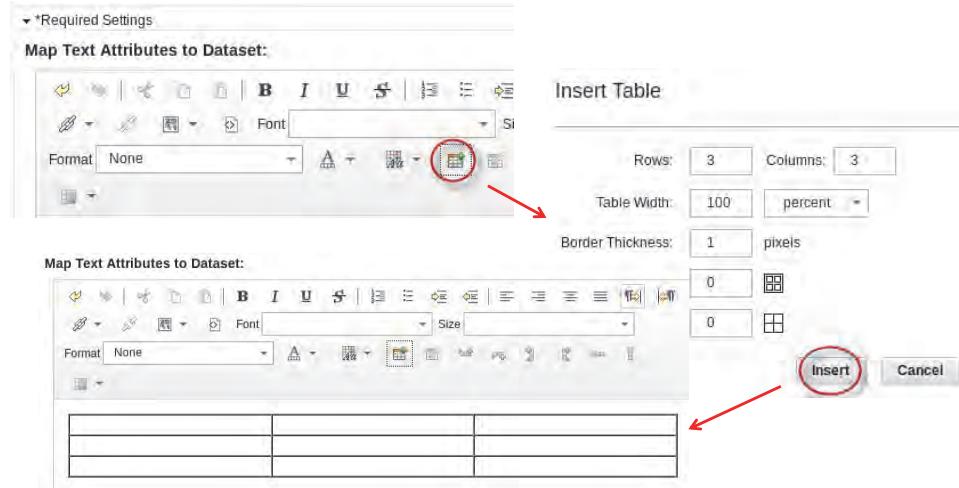


Inserting data set tags

When the required settings and map text attributes to dataset are opened, it prompts for a dataset to be selected from the right side of the HTML editor section. All of the parameter names that you can use are listed. Clicking an item on the right side generates dynamic text on the left side in the editor.

Adding tables

- Click the **Insert Table** icon, specify the **row and column count**, and select **Insert**

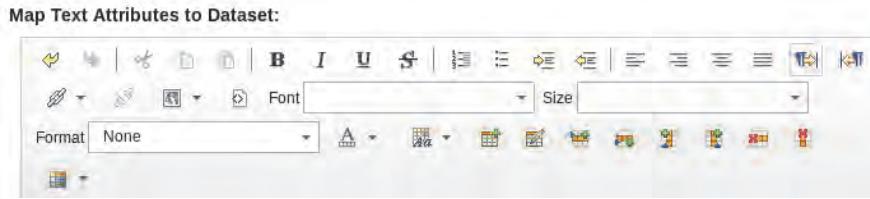


Adding tables

Tables can be added using the HTML editor. You have other options to use like table width, border thickness, font, size, format, and other edit options.

Formatting smart text content

- Control widget or table cell background and foreground color
- Control font, font size, style, color, background color, and alignment



Formatting smart text content

You have complete control over how you want the widget to appear.

Instructor demonstration

Using hotspot widgets with web widgets



Exercise 4

Using hotspot widgets with web widgets

Exercise

Complete Unit 4 Exercise 4 in the exercise guide.



Note: You can use the CPU_AVERAGES dataset in the exercises to reduce the quantity of managed systems in the table. This results in a single entry per managed system. This eliminates multiple entries for each Managed System name caused by having multiple CPUs for each managed system.

Unit Summary

You now should be able to perform the following tasks:

- Describe how to configure and use the Jazz for Service Management web widget in business dashboards
- Use debugging tools to analyze and examine event parameters from dashboard widgets
- Substitute NodeClickedOn and displayURL widget event parameter values in web widget addresses to send context information
- Use web widgets to add non-IBM application consoles to dashboard pages
- Use hotspot widgets to change the data that is shown in web widgets and create interactive dashboard pages
- Use the SmartText widget to dynamically create and serve HTML text and data set data

Unit 6 Integrating Netcool/OMNibus with DASH

IBM Training



Unit 6 Integrating Netcool/OMNibus with DASH

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this unit, you learn how to integrate Netcool/OMNibus event data with DASH dashboards. You learn how to use Netcool/OMNibus data filters to use in DASH data sets. You also learn how to troubleshoot data communication problems between the DASH server and the Netcool/OMNibus UI data provider.

Objectives

When you complete this unit, you can perform the following tasks:

- Describe the Netcool/OMNibus monitoring architecture
- Describe and use WebGUI UI data provider data sets
- Configure dashboard widgets to show Netcool/OMNibus event data
- Create transient event filters with web-based event viewers and Active Event Lists
- Troubleshoot the Netcool/OMNibus WebGUI UI data provider

Lesson 1 OMNibus overview

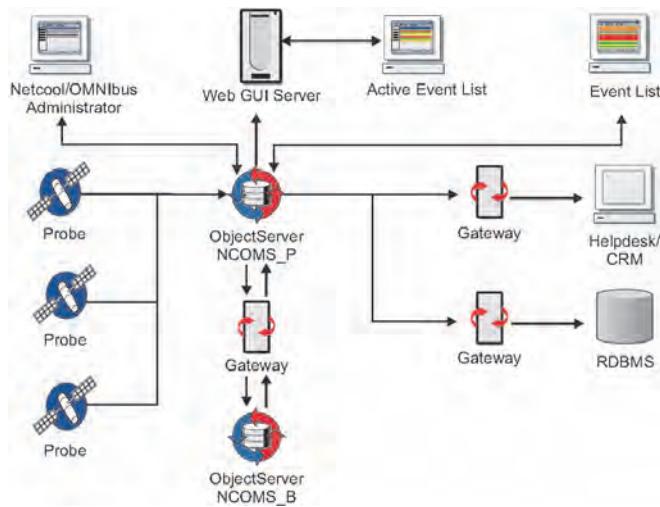
IBM Training



Lesson 1 OMNibus overview

Deployment architecture

- OMNibus Web GUI 8.1 is integrated with and requires a DASH server
- Application management is done in the DASH console
- Event data is available with AEL and event viewer widgets



Deployment architecture

The figure shows an overview of the Tivoli Netcool/OMNibus component architecture.

- Probes connect to an event source, detect and acquire event data, and forward the data to the ObjectServer as events.
- The ObjectServer is the in-memory database server at the core of Tivoli Netcool/OMNibus.
- Gateway replicates alerts in another ObjectServer in a failover configuration.
- Alerts that are sent to the ObjectServer can be viewed in events lists in the Web GUI or the desktop.
- Web GUI is a web-based application that presents event data from multiple data sources in various graphical formats in supported web browsers and mobile devices. The Web GUI includes most features of the Tivoli Netcool/OMNibus desktop components. You can create dashboards in Dashboard Application Services Hub and add the widgets for displaying the event data. The event feed for Web GUI widgets is provided by the data source or data sources.
- Widgets from the Tivoli Widgets Library (TWL) obtain event data from the Web GUI data provider.
- More gateways are also configured to forward alerts to other applications, such as a helpdesk or Customer Relationship Management (CRM) system, and a relational database management system (RDBMS).
- Administrator tools - can be used to configure and manage the system.
- Desktop tools - The desktop is an integrated suite of graphical tools used to view and manage events, and to configure how event information is presented.

- Administration tools: Tivoli Netcool/OMNibus includes tools that administrators can use to configure and manage the system.
- Netcool MIB Manager: Netcool MIB Manager is an IBM® Eclipse-based application that you can use to parse Simple Network Management Protocol (SNMP) management information base (MIB) files, from which you can generate Netcool rules files.
- About probes: Probes and Telco Service Managers (TSMs) connect to an event source, detect and acquire event data, and forward the data to the ObjectServer as alerts. TSMs operate in the same manner as probes but have some additional functions. Probes and TSMs use the logic specified in a rules file to manipulate the event elements before converting them into fields of an alert in the ObjectServer alerts.status table.

Getting started

- Default users and groups are created when WebGUI is installed
 - Members of Netcool_OMNIbus_Admin can create, delete, and modify events, OMNIbus gauge widgets, and WebGUI tools and filters
 - Members of Netcool_OMNIbus_User can view events in event viewer and AEL widgets
- Consider implications of a hard-coded user ID (such as ncoadmin) in the connection document configuration:
 - Any DASH effectively inherits the roles that are assigned to the connection document user ID, such as:
 - System filters for an administrative user
 - User filters
 - OMNIbus events by using the proxy user's restriction filter

Group Name	Roles assigned	Default user
Netcool_OMNIbus_User	ncw_user, netcool_ro	ncouser
Netcool_OMNIbus_Admin	ncw_admin, ncw_dashboard_editor, ncw_gauges_editor, ncw_user, netcool_rw	ncoadmin

Getting started

This slide shows the groups, users, and roles assigned in a default installation.

Lesson 2 WebGUI UI data provider data sets

IBM Training

IBM

Lesson 2 WebGUI UI data provider data sets

Provider and data sources overview

- Provider: Netcool/OMNibus Web GUI
- Data sources:
 - Aggregated data:** Data sets that contain high-level OMNibus data metrics
 - All data:** Data from all other provider data sources
 - Events:** Data sets that contain lists of events
 - Event information:** Data sets that contain detailed information for a single event

Provider: Netcool/OMNibus Web GUI > Datasource: Aggregated data

Event Grouping	Groups of events for Netcool/OMNibus Web GUI
Event Summary	Event counts by severity for a specified OMNibus event
Filter Summary	Event statistics for a specified OMNibus filter
Metrics	Current values for specified OMNibus metrics

Provider: Netcool/OMNibus Web GUI > Datasource: Events

Events: Default (global)	Netcool/OMNibus Web GUI events for the 'Default' global view
Events: RawEvents (system)	Netcool/OMNibus Web GUI events for the 'RawEvents' system view
Events: SelfMonitoring (system)	Netcool/OMNibus Web GUI events for the 'SelfMonitoring' system view
Events: ServiceState (system)	Netcool/OMNibus Web GUI events for the 'ServiceState' system view
Events: SLAStatus (system)	Netcool/OMNibus Web GUI events for the 'SLAStatus' system view

Provider: Netcool/OMNibus Web GUI > Datasource: Event information

Event Details	Event details for a specified OMNibus event
Event Fields	Event field values for a specified OMNibus event

© Copyright IBM Corporation 2016

Provider and data sources overview

7

These are the data sets that you will see with Netcool OMNibus.

Events data set: Events data source

- This data source contains all events for a specified Web GUI filter
- The data provider contains the Events data set per Web GUI view
 - A nonadministrative user sees all global and user views
 - Web GUI filters or views are defined in the DASH console
 - Example data set is Events: Default (global)
- This data source has the following parameters:
 - Filter (mandatory): Web GUI filter to select event subset
 - Data source (optional): Web GUI data source
- Sample data is shown in the following table

Serial	Node	Ack	Summary	Count
45	sol-dashsvr1	Yes	Memory exhausted	1
48	sol-dashsvr1	No	Due for reboot	3
49	win-svr10	No	Administrator logged in	3

Events data set: Events data source

This dataset contains all of the events for a specified filter.

- Serial numbers are unique event numbers.
- Events can be acknowledged to show the ownership of an event.
- Summary is the description.
- The count field gives you a number of the particular event type rather than produce multiple events of the same alert.

Filter Summary data set: Aggregated data data source

- This data source contains high-level statistics for a specified Web GUI filter
 - All data is in a single row
 - Example data includes total event count, metric, maximum severity, and event counts for each severity
- This data source has the following parameters:
 - Filter (mandatory): Web GUI filter whose high-level data is required
 - Data source (optional): Web GUI data source
- Sample data is shown in the following table

Filter Name	Filter Type	Total Event Count	Metric	Maximum Severity	Severity 5 Event Count	Severity 4 Event Count
USA	global	19	2	5	2	5

Filter Summary data set: Aggregated data data source

This example shows the total of the various event counts.

Event Summary data set: Aggregated data data source

- This data source contains a severity distribution for a specified Web GUI filter
 - Contains one row per severity level
 - Each row indicates the number of events that match that severity
- This data source has the following parameters:
 - Filter (mandatory): Web GUI filter whose events will be summarized
 - Data source (optional): Web GUI data source
- Sample data is shown in the following table

Severity Name	Event Count
Critical	20
Major	76
Minor	91
Warning	33

Event Summary data set: Aggregated data data source

This example shows one row per severity type and the total number of events in the event count field for each severity.

Event Grouping data set: Aggregated data data source

- This data source contains custom event distributions for a Web GUI filter
 - Like Event Summary, but grouping by custom fields, not just severity
 - Multiple levels of grouping are possible; for example, location plus acknowledged
 - Rows are hierarchical and not suited to display in flat table
- This data source has the following parameters:
 - Filter (mandatory): Web GUI filter whose events will be grouped
 - View (recommended): Web GUI view, which defines the fields to group by
 - Data source (optional): Web GUI data source
- Sample data is shown in the following table

Group	Count	Sev
All	90	Critical
UK	45	Critical
Yes	40	Warning
No	5	Critical

Event Grouping data set: Aggregated data data source

In this hierarchical display, All contains a count for everything. Under All, events might be broken out by countries. Under a country, there might be Yes or No breakouts.

Event Fields data set: Event information data source

- This data source contains all field values for a single OMNIbus event
 - All field values from alerts.status ObjectServer table are returned
 - Same information as used for first tab in AEL information dialog box
- This data source has the following parameters:
 - Serial (mandatory): Serial ID of event whose field values are needed
 - Data source (mandatory): Web GUI data source to obtain data from
- Sample data is shown in the following table

Field	Value
Summary	Login failed
Node	jupiter
Severity	Major
AlertGroup	webtop

Event Fields data set: Event information data source

This example shows some fields for a specific event.

Event Details data set: Event information data source

- This data source contains all detail attributes for a single OMNIbus event
 - All relevant entries from alerts.details ObjectServer table are returned
 - Same information as used for second tab in AEL information dialog box
- This data source has the following parameters:
 - Serial (mandatory): Serial ID of the event whose details are needed
 - Data source (mandatory): Web GUI data source
- Sample data is shown in the following table

Field	Value
Attr1	custom1
Attr2	custom2

Journal Entries data set: Event information data source

- This data source contains all journal entries for a single OMNIbus event
 - All relevant entries from alerts.journal ObjectServer table are returned
 - Same information as used for third tab in AEL information dialog box
- This data source has the following parameters:
 - Serial (mandatory): Serial ID of event whose journal entries are needed
 - Data source (mandatory): Web GUI data source to obtain data from
- Sample data is shown in the following table

User ID	Date/Time	Journal Entry
lukather	Thu Nov 7 10:15:13 2013	Alert acknowledged by lukather
lukather	Thu Nov 7 10:16:00 2013	Alert prioritized from Major to Minor by lukather
paich	Fri Nov 8 09:31:15 2013	Check tool run by paich

Journal Entries data set: Event information data source

Journal entries display a dialog in the posts, by which operators have taken some action based on the alert. This example shows the information for a single event.

Lesson 3 Using DASH widgets with Netcool/OMNibus event data

IBM Training

IBM

Lesson 3 Using DASH widgets with Netcool/OMNibus event data

Populating DASH widgets from OMNIbus Web GUI provider

- Specific data sets are best used with specific widgets
- Use depends on nature and shape of the data

Populating DASH widgets from OMNIbus Web GUI provider

You can use OMNIbus data sources and data sets which are provided by OMNIbus.

Events that match a specific condition

- The widget shows the number of events that match a particular condition in a gauge or volume bar
- Example: Number of unacknowledged events, in an analog gauge
- Widget configuration:
 - Dataset = Metrics
 - Metrics = Old Unacknowledged
 - Value = Value
 - Data Sources = OMNIBUS



Events that match a specific condition

This example shows that 23 old, unacknowledged events are currently on the system. The number of 23 events is considered a warning.

Maximum severity of events

- The widget shows the maximum severity of events that match a specific condition in a gauge
- Example: Maximum severity of all US events, in a status gauge
- Widget configuration:
 - Dataset = Filter Summary
 - Value = Maximum Severity
 - Filter = USA (global)



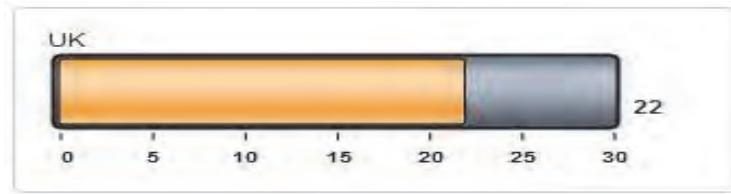
- Note: DASH status does not match OMNIbus severity exactly

Maximum severity of events

A company can have multiple offices around the world. The highest severity in the North America office is critical while the highest severity in Europe is a warning.

Set of events in a gauge or volume bar

- The widget shows a metric for a set of events that match a specific condition in a gauge or volume bar
- Example: Summed tally for all UK events, in a volume bar
- Widget configuration:
 - Dataset = Filter Summary
 - Value = Maximum Severity
 - Filter = UK (global), defined with Sum(Tally) as the metric



Set of events in a gauge or volume bar

On the right side of the gauge is 22, which is the total number of events. The orange bar reaches 22.

Set of fields for selected events

- The widget shows a specific set of fields for all events that match a specific condition in a table or list
- Example: Serial, NodeAlias, and Location for all unacknowledged events, in a table
- Widget configuration:
 - Dataset = Events: Locations (global)
 - Locations (global) view is defined with Serial, NodeAlias, Location fields
 - Filter = Unacknowledged (global)

Unacknowledged Events		
Serial	NodeAlias	Location
128	tigris	Canada
163	ganges	Canada
164	tigris	Canada
111	polar	India
440	nando	India

Total: 76 Selected: 0

Set of fields for selected events

The table widget is used to display all of the unacknowledged events. The NodeAlias in this example might be named after servers or companies in the locations.

Field values, details, or journal entries for a single event

- The widget shows all field values, details, or journal entries for a single event in a table
- Example: All journal entries for an event, in a table
- Widget configuration:
 - Dataset = Journal Entries
 - Serial = 0
 - Data source = NCOMS
- Page configuration:
 - Wire from Events table to Journal Entries table, using NodeClickedOn

Unacknowledged Events		
Serial	NodeAlias	Location
98	winbox1	USA
99	winbox3	USA
100	winbox2	USA
102	devX	USA

Total: 76 Selected: 1

Journal		
User ID	Date/Time	Journal Entry
ncoadmin	Mon Nov 11 12:34:38 GMT 2013	Alert acknowledged by ncoadmin.
ncoadmin	Mon Nov 11 12:34:58 GMT 2013	Alert prioritized from Minor to Major by ncoadmin.
ncoadmin	Mon Nov 11 12:35:06 GMT 2013	Ownership of alert taken by ncoadmin.

Total: 3 Selected: 0

Field values, details, or journal entries for a single event

Two table widgets are displayed. One is a repeat of the unacknowledged events. The right table contains journal entries. A wire is made between the unacknowledged events so that one event can be clicked and the journal entries for that event are displayed.

Events grouped by multiple fields

- The widget shows the number of events that match a particular condition, grouped by multiple fields, in a tree table
- Example: All unacknowledged events, grouped by Location and Node
- Widget configuration:
 - Dataset = Event Grouping
 - Filter = Unacknowledged (global)
 - View = Offices (global)
 - Offices (global) view is defined with Location, Node as grouping fields

Group	Count	Sev
All	76	Critical
Canada	3	Minor
13.140.177.3	1	Minor
13.140.177.7	2	Minor
India	8	Major
UK	19	Minor

Events grouped by multiple fields

This example is showing filtering of the hierarchy. All represents all of the events. Clicking the plus signs breaks out the countries and their totals. Expanding each country is showing IP Addresses.

Lesson 4 Transient filters and dynamic event lists

IBM Training



Lesson 4 Transient filters and dynamic event lists

Creating filtered event lists with context events

- You can configure dashboards to show dynamically filtered event lists, using NodeClickedOn events and OMNibus *transient filters*
- You use event parameters and parameter substitution with a web widget-based event viewer
- Example:
 - A dashboard page contains a table widget and an event viewer widget
 - Clicking a row in the table triggers a filter in the event viewer, showing only related events

The screenshot shows a dashboard interface. On the left, there is a table titled "Table" with three rows: Asia (Good), Europe (Good), and US (Bad). The "Europe" row is highlighted with a blue selection bar and has a red arrow pointing from it to the right side of the screen. On the right, there is a window titled "Events for Selected TBSM Service" with the sub-header "SelectedEvents1". This window displays a list of events for the selected service. The table data is as follows:

Sev	Ack	Node	Alert Group
Info	No	EuropeBigBux	

Integrating Netcool/OMNibus with DASH

24

© Copyright IBM Corporation 2016

Creating filtered event lists with context events

In this example with the table on the left, the display name is representing regions. When a region is selected, like Europe, the table on the right displays all of the Europe related events.

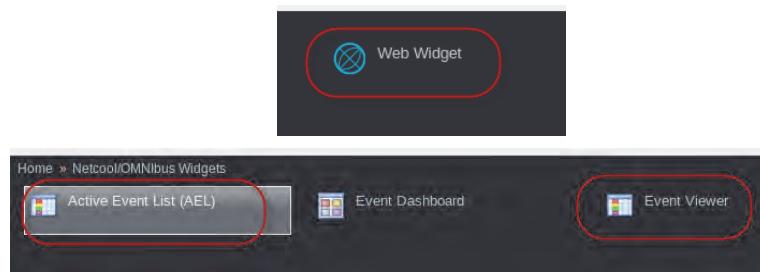
OMNibus transient filters

- Transient filters are OMNibus event filters that are defined as part of a command script or in a URL that opens an event viewer or Active Event List
- Transient filters are only valid during the time that the user is logged on to the DASH or WebGUI console
- The filter applies SQL statements against the alerts.status table in the ObjectServer database to create a filtered set of events

Configuration of NCMS on dash151.pocibm.com:4100					
		Databases, Tables and Columns			
		Column Definitions		Data View	
Name /	Data Type	Le...	Pri...	No...	
Acknowledged	Integer	4	X	X	
Agent	VarChar	64	X	X	
AlertGroup	VarChar	255	X	X	
AlertKey	VarChar	255	X	X	
BSM_ClassName	VarChar	512	X	X	
BSM_Identity	VarChar	10...	X	X	
BSM_SubIdentity	VarChar	10...	X	X	
Class	Integer	4	X	X	
Customer	VarChar	64	X	X	
EventId	VarChar	255	X	X	
ExpireTime	Integer	4	X	X	
ExtendedAttr	VarChar	40...	X	X	
FirstOccurrence	UTC	4	X	X	
Flash	Integer	4	X	X	
Grade	Integer	4	X	X	
Identifier	VarChar	255	✓	✓	
InternalLast	UTC	4	X	X	

Using event viewers in a web widget

- The Active Event List and Event Viewer widgets can be added to dashboard pages from the widget palette or through a URL in a web widget
- Only preconfigured event filters can be used with the Active Event List and Event Viewer widgets
- Using a URL in a web widget allows you to specify and apply a transient filter when the event viewer is opened
- You use parameter substitution in the URL so that the transient filter can dynamically respond to NodeClickedOn events



Using event viewers in a web widget

The Netcool/OMNibus product provides some widgets.

A web widget can be used with parameter substitution.

Opening event viewers in a web widget

- Using a URL to open an event viewer:

- Event viewer:

`https://<dashHost>:<port>/ibm/console/webtop/eventviewer/eventViewer.jsp?<queryString>`

- Active Event List:

`https://<dashHost>:<port>/ibm/console/webtop/AELView?<queryString>`

Opening event viewers in a web widget

This slide shows how the OMNIbus event viewer and OMNIbus Active Event List can be displayed in a web widget.

Event viewer URL query strings

- The general form of the query string, for both event viewer types:
`sql=<filterSQL>&transientname=<filterName>&viewname=<viewName>&viewtype=<viewType>&datasource=<datasourceName>`
 - “**<filterSQL>**”: The transient filter SQL (must be surrounded by double quotation symbols)
 - Example: "Node='Server1' AND Severity=5"
 - **<filterName>**: A text string that identifies the transient filter
 - **<viewName>**: The name of a predefined view to be applied to the viewer
 - **<viewType>**: The type of view, such as global or system
 - **<datasourceName>**: The ObjectServer data source name
 - To see available data sources, select the **Event Administration > Data Sources** task
- Use parameter substitution in the filter string to respond to NodeClickedOn events
- Query string parameter=value pairs are separated by the **&** symbol

Event viewer URL query strings

This slide shows the parameters being passed as the transient which are used by the web page.

Example: Showing events related to a table selection

- In this example, the dashboard page contains two widgets:
 - Table widget: Shows a list of Tivoli Business Services Manager services
 - Web widget: The home page URL is configured to show all events that have a Node value that corresponds to the TbsmServiceInstanceName in the table widget NodeClickedOn event
- Using event substitution, the web widget URL is:
<https://dash151.poc.ibm.com:16311/ibm/console/webtop/eventviewer/eventViewer.jsp?sql='Node='%%TbsmServiceInstanceName%%'&transientname=AllSelected&viewname=Default&viewtype=global&datasource=OMNIBUS>

The screenshot shows a dashboard with two main components. On the left is a 'Table' widget with three rows: Asia (Good), Europe (Good), and US (Bad). The 'Europe' row is highlighted with a blue dashed border. An arrow points from this highlighted row to the right side of the screen. On the right is a 'Events for Selected TBSM Service' list. The list has a header row with columns: Sev, Ack, Node, and Alert Group. Below the header, there are five rows of event data, all of which have 'Node' values corresponding to 'Europe'. The 'Alert Group' column is empty.

Display Name	TBSM Status
Asia	Good
Europe	Good
US	Bad

Sev	Ack	Node	Alert Group
Info	No	EuropeBigBux	

Example: Showing events related to a table selection

In this slide, note the length of the URL and how it is used to display the transient events with the substitutions.

Lesson 5 Troubleshooting the WebGUI UI data provider

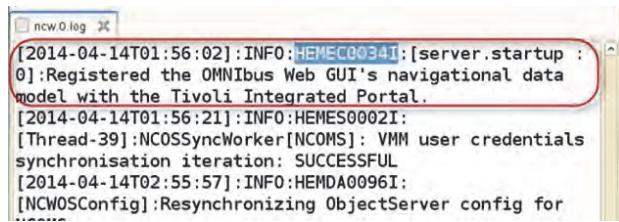
IBM Training



Lesson 5 Troubleshooting the WebGUI UI data provider

Checking Web GUI data provider registration

- Ensure that the provider is successfully registered with Tivoli Integrated Portal
- On the DASH server, change directory to \$JazzHOME/profile/logs/ncw
- Open ncw.0.log and search for the following entry:
HEMEC0034I: Registered the OMNIBus Web GUI's navigational data model with the Tivoli Integrated Portal



```
[2014-04-14T01:56:02]:INFO:HEMEC0034I:[server.startup : 0]:Registered the OMNIBus Web GUI's navigational data model with the Tivoli Integrated Portal.  
[2014-04-14T01:56:21]:INFO:HEMES0002I:  
[Thread-39]:NCOSyncWorker[NCOMS]: VMM user credentials synchronisation iteration: SUCCESSFUL  
[2014-04-14T02:55:57]:INFO:HEMDA0096I:  
[NCWOSConfig]:Resynchronizing ObjectServer config for
```

Checking Web GUI data provider registration

There are other OMNIBus courses available. This course just shows you some basic troubleshooting.

The Netcool/OMNIBus provider must be successfully registered with Tivoli Integrated Portal.

Checking the DASH connection with the ObjectServer

- Ensure that Web GUI is connected to ObjectServer database
 - Example log entry: HEMDA0254I:[NCWBringUpWorker]:Connection jdbc:sybase:Tds:sol-dashsvr1:4100 is established. There are 1 established connections in the connection pool.
 - Look for HEMDA0265E:[NCWBringUpWorker]:Failed to initialize connection pool for jdbc:sybase:Tds:sol-dashsvr1:4100. The server may be down.

```
0]:Can not schedule any task. The limerControl is disabled.  
[2014-04-14T01:55:59]:INFO:HEMDA0206I:  
[NCWBringUpWorker]:Connecting to  
jdbc:sybase:Tds:tbsm01.tivoli.edu:4100.  
[2014-04-14T01:55:59]:INFO:HEMDA0254I:  
[NCWBringUpWorker]:Connection  
jdbc:sybase:Tds:tbsm01.tivoli.edu:4100 is established.  
There are 1 established connections in the connection pool.  
[2014-04-14T01:55:59]:INFO:HEMDA0206I:  
[NCWBringUpWorker]:Connect  
jdbc:sybase:Tds:tbsm01.tiv  
[2014-04-14T01:55:59]:INFO:  
[NCWBringUpWorker]:Connect  
jdbc:sybase:Tds:tbsm01.tiv  
There are 2 established co  
[2014-04-14T01:55:59]:INFO:  
ncw0.log X  
to the ObjectServer jdbc:sybase:Tds:tbsm01.tivoli.edu:4100.  
[2013-12-17T17:55:34]:SEVERE:HEMDA0265E:[NCWBringUpWorker]:Failed to  
initialize connection pool for jdbc:sybase:Tds:tbsm01.tivoli.edu:4100. The  
server may be down.  
[2013-12-17T17:55:34]:INFO:[NCWBringUpWorker]:Trying reconnect to  
jdbc:sybase:Tds:tbsm01.tivoli.edu:4100 in 10 seconds.  
[2013-12-17T17:55:43]:WARNING:HEMES0003W:[Thread-37]:NCOSSyncWorker[NCOMS]
```

Checking the DASH connection with the ObjectServer

This example illustrates whether the connection is established or failed to establish with the OMNibus connection pool.

Diagnosing problems

- Check Web GUI logs on the DASH server
 - Logs are in \$JazzHOME/profile/logs/ncw
 - Most data provider messages have message ID HEMDP
- A common problem is that the connecting user does not have the **ncw_admin** or **ncw_user** role

Diagnosing problems

If your OMNIbus system will not start, list the contents of the **/opt/IBM/tivoli/netcool/omnibus/var** directory for the existence of .pid files. If OMNIbus tries to start and sees that these files exist, it will not start. This is typically caused by an improper shutdown of an operating systems or OMNIbus system. Remove these .pid files and restart OMNIbus.

Exercise 1 Adding event data to a dashboard page

Exercises

Complete Unit 6 Exercise 1 from the exercise guide. This is the only exercise for this unit.

Summary

Now that you have completed this unit, you can perform the following tasks:

- Describe the Netcool/OMNibus monitoring architecture
- Describe and use WebGUI UI data provider data sets
- Configure dashboard widgets to show Netcool/OMNibus event data
- Create transient event filters with web-based event viewers and Active Event Lists
- Troubleshoot the Netcool/OMNibus WebGUI UI data provider

Unit 7 Integrating IBM Tivoli Monitoring and DASH

IBM Training



Unit 7 Integrating IBM Tivoli Monitoring and DASH

© Copyright IBM Corporation 2016
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this unit, you learn the IBM Monitoring dashboard integration points. You learn how to retrieve monitoring agent data from the UI data provider and use that data in several types of dashboard widgets. You learn how the system status and health dashboards can be implemented to provide quick operational dashboards that provide an enterprise-wide view of your monitoring environment.

Objectives

- When you complete this unit, you can perform the following tasks:
- Describe the Tivoli Monitoring architecture
- Enable the Tivoli Monitoring data provider
- Explain how the dashboard data requester connects to IBM Tivoli Monitoring 6.3
- Use the Tivoli Monitoring data provider with widgets
- Create a custom dashboard

Lesson 1 IBM Tivoli Monitoring overview

IBM Training



Lesson 1 IBM Tivoli Monitoring overview

3

© Copyright IBM Corporation 2016

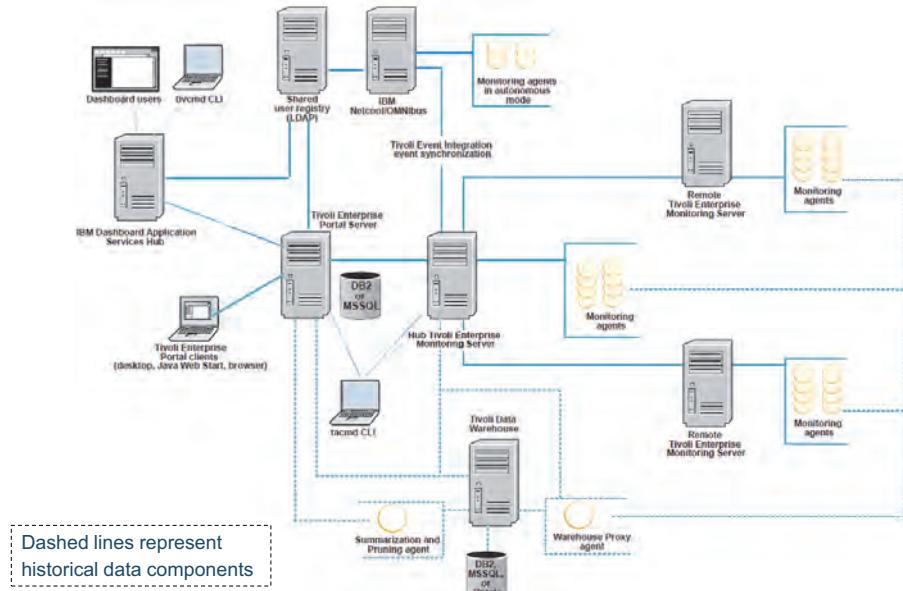
Enterprise monitoring

- Enterprise monitoring is designed to manage resources that are important for your daily business operations
- Enterprise monitoring data might be any of the following types:
 - Incidents that indicate potential problems that require attention
 - Real-time statistics about a specific resource or application
 - Historical statistics that provide insight into the long-term behavior of components
- You can use IBM Tivoli Monitoring for these tasks:
 - View availability and performance data that provides insight into the components, applications, and services in your enterprise
 - Receive notifications of incidents that require attention
 - Diagnose problems

Enterprise monitoring

The monitoring product has evolved over many years.

IBM Tivoli Monitoring components



IBM Tivoli Monitoring components

Monitoring Server: the heart of the operation

Remote Monitoring Servers: offload cycles from the Monitoring Server

Agent: typically collect monitoring data

Portal Server and database: presentation services, maintaining accounts and roles

Portal Clients: console interfaces for viewing the monitoring information

Tivoli Data Warehouse, agents, and database: collect, process, and maintain data used for historical reporting

LDAP: secure method for validating IDs and passwords

Netcool/OMNIbus: event processing and displaying

Autonomous agents: agents collecting monitoring information that OMNIbus uses, and are not required to connect to a monitoring server

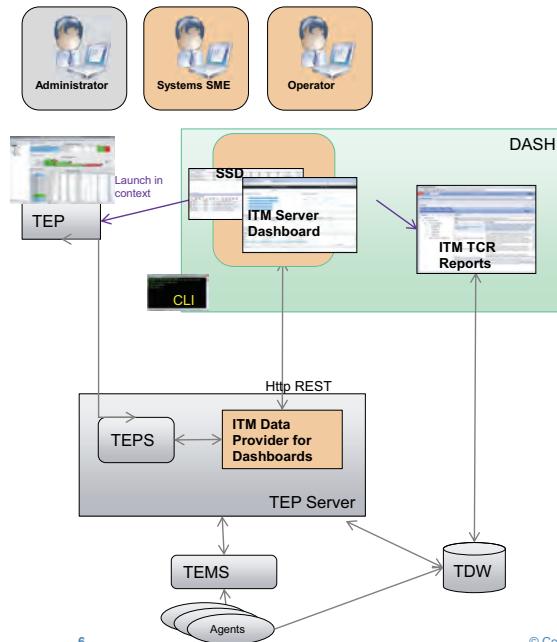
Dashboard Application Services Hub: the application to create pages with widgets that can display monitoring data

Dashboard users: the console operators of the dashboard application

CLI interfaces itmcmd and tacmd: the functions provided for issuing commands from a terminal or command interface

ITM user interface strategy

- Primary persona
 - Operator
 - LOB's Owners
- Secondary persona
 - Systems SME, Administrator
- ITM Server Dashboards
- Self Service Dashboards
- ITM Data Provider for Dashboards
- Policy Server CLI



Integrating IBM Tivoli Monitoring and DASH

6

© Copyright IBM Corporation 2016

ITM user interface strategy

The primary consumers are the operators and managers. Those consumers open the dashboard on their device, check conditions, and move on with their day.

Administrators require a deeper drill capability to see more information. The Tivoli Enterprise Portal Server formats the requested data and displays the data to the user.

When the product is fully installed, there are server dashboards for users to obtain an immediate view of the availability of components. You can also create dashboards for your own purposes. How you create those dashboards is through the ITM data provider.

The policy server helps with security and control over what monitoring data users can access.

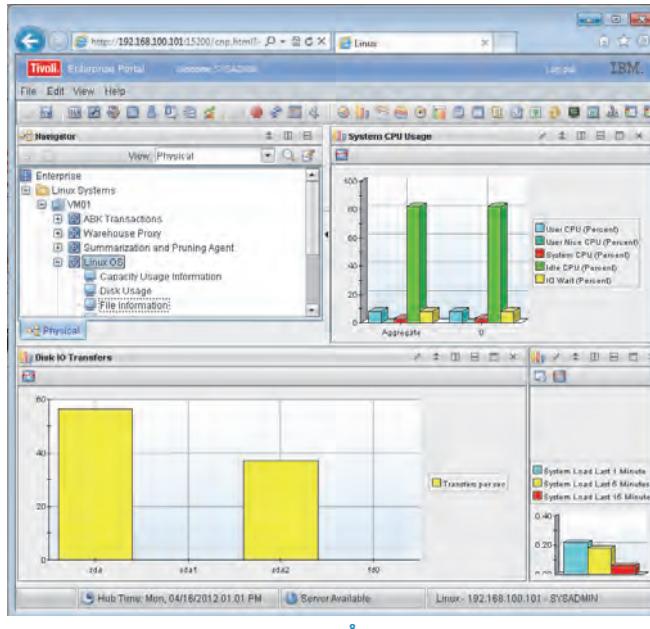
Tivoli Enterprise Monitoring Agents

- Collect data from managed systems
- Can be local or remote, depending on monitored system
- Store collected values as attributes in attribute groups to simplify managing large amounts of data
- Monitor for problems and can issue system commands
- Can store short-term historical data
- Can be one of three types
 - Operating System (OS)
 - Application
 - Specialized
- Are available on z/OS, UNIX, Linux, Linux on System z, i5, and Windows

Tivoli Enterprise Monitoring Agents

Agents are primarily installed to collect monitoring information. There are some agents that provide a service instead of collecting monitoring information.

Portal Java client



Portal Java client

There are three client interfaces. This example is showing the Java portal client. The other two are the desktop client and the web client.

This screen is referred to as a workspace. In the upper left part of the slide is the navigator. Each navigator entry can be clicked to display another workspace. The navigator displays a hierarchical list of navigators showing each of the components in the managed environment.

Each workspace has one or more views associated with the workspace. Most workspaces and views are product provided, but you can also create your own workspaces and views.

Key requirements for UI data provider data access

- Requires minimum IBM Tivoli Monitoring version 6.3 (TEMS&TEPS only)
- Highly responsive web user interface, with product-provided usable content for Monitoring and Problem Determination scenarios
- Incorporate best practices for Server monitoring
- Visualize Managed Systems and Groups with corresponding events and metrics
- Provide an interface to access data for visualization from ITM infrastructure (Agents, TEMS, TEPS, and Warehouse)
- Ability to refine Tivoli Monitoring data for visualization
- Enable granular data authorization for role based access control

Key Requirements for UI data provider data access

Users can just update the Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server to 6.3 and leave the agents at older versions for a quick time to value. That was one of the key ideas behind the design of the Dashboard Provider development.

IBM Tivoli Monitoring ships with provided dashboards that display some of the key performance metrics and event information, and incorporate some best practices.

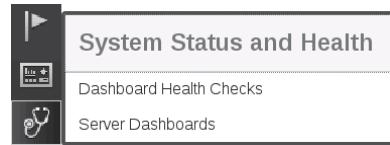
The ability to refine IBM Tivoli Monitoring data for visualization is key function.

The ability to refine and build some key dashboards with deep dive metrics is a HUGE win for an administrator with the user community.

Both the Tivoli Enterprise Portal and the IBM Tivoli Monitoring Data Provider use the application's ODI file, which describes all of the attribute groups and attributes that the agent can support

System status and Health dashboards

- Ready to use as provided
- Incorporates best practices for IT Operations and LOB personas
- Features
 - Visualize health (availability & performance)
 - Identify problems and trends (Problem determination)
 - Isolate systems with problems
 - Monitoring utilization of system resources
- Complementary to TEP
 - Provides finer grained authorization control
 - Provides contextual launch into TEP and other tools
 - Targeted to non administrators

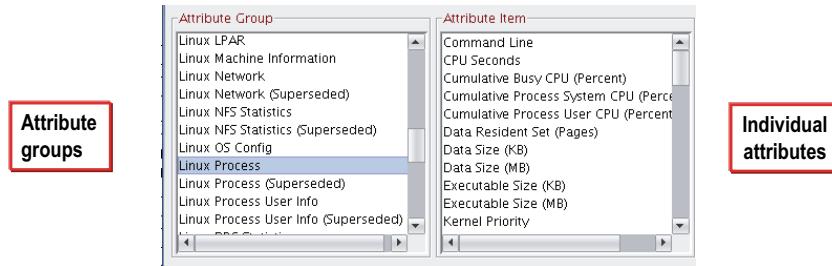


System Status and Health dashboards

The System Status and Health is available if you install all of the components for the Dashboard environment.

Attributes

- Attributes are data items that the agents collect and send to the portal client for display or event generation
- When creating a situation, use one or more attributes in your condition
- When you evaluate the situations, actual values replace the attributes
- The resource assigns individual attributes logically into attribute groups



Attributes

Attributes are the basic element of information that monitoring agents collect. Attributes are collected into attribute groups.

Query

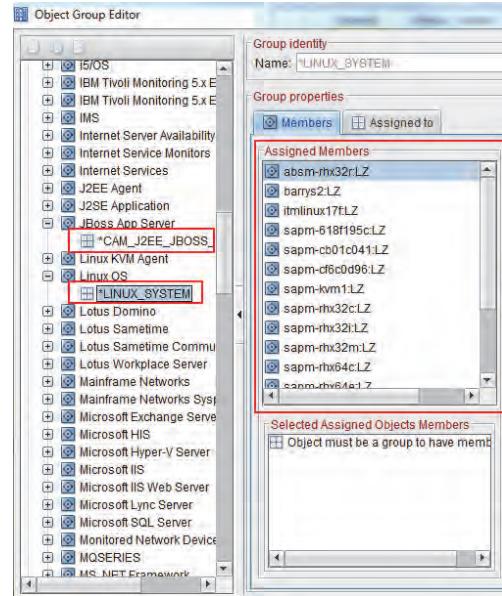
- A query defines what data is requested from a Tivoli Enterprise Monitoring Agent or other data source when a workspace is accessed or refreshed
- Queries are used only for table views and chart views
- Queries are driven by demand from the portal client
 - Initially accessed
 - Workspace refresh
- Queries form the basis for the ITM UI data provider data sets
- IBM provides a set of predefined queries for retrieving all data that is collected by Tivoli Enterprise Monitoring Agents
- Queries can include data from Tivoli Enterprise Monitoring Agents or other ODBC data sources
- You can write queries against database tables
- When you issue a query, you filter at the source

Query

Queries are requests to obtain data. When a workspace is clicked, a query is dispatched to request information to be put into views.

Groups

- Users can aggregate data for widgets by the groups defined inside ITM.
 - i.e. *LINUX_SYSTEM is every system that has a Linux OS agent on it.
- Users can create custom groups inside ITM to help aggregate data through the ITM Object Group Editor.
- Use Groups to retrieve data if you are not using Impact to sort and arrange data.



Groups

An object group editor is used to create and maintain groups. There are also built-in groups that are product provided.

Lesson 2 Connecting Tivoli Monitoring for dashboards

IBM Training



Lesson 2 Connecting Tivoli Monitoring for dashboards

14

© Copyright IBM Corporation 2016

IBM Tivoli Monitoring UI data provider

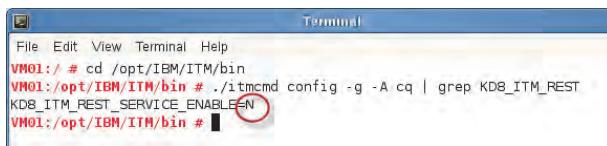
- The UI data provider exposes monitoring agent attributes
 - Includes historical data if configured with Tivoli Data Warehouse
- You can optionally install the dashboard data provider during the Tivoli Enterprise Portal Server configuration
- Dashboard widgets read-only data from the hub monitoring server and monitoring agent
 - The IBM Infrastructure Management Dashboards for Servers component uses the UI data provider to pull management data into the dashboards

IBM Tivoli Monitoring UI data provider

You can access data sources and data sets that contain the IBM Tivoli Monitoring attribute information.

IBM Tivoli Monitoring dashboard data provider (continued)

- IBM Tivoli Monitoring V6.3 includes the Infrastructure Management Dashboards for Servers
 - It shows data for the OS agents
 - These server dashboards use the dashboard data provider to retrieve data
- You must configure a connection to the dashboard data provider in the Dashboard Application Services Hub
- For more information, see Preparing your dashboard environment and Creating a connection to the IBM Tivoli Monitoring dashboard data provider
- Set the portal server environment variable KDB_ITM_REST_SERVICE_ENABLE to Y



A screenshot of a terminal window titled "Terminal". The window contains the following text:
File Edit View Terminal Help
VM01:/ # cd /opt/IBM/ITM/bin
VM01:/opt/IBM/ITM/bin # ./itmcmd config -g -A cq | grep KDB_ITM_REST
KDB_ITM_REST_SERVICE_ENABLE=EN
VM01:/opt/IBM/ITM/bin #

Below the terminal window, the command is repeated:
Enable the dashboard data provider ? (1=Yes, 2=No)(Default is: 2): 1
Enter the Domain override:

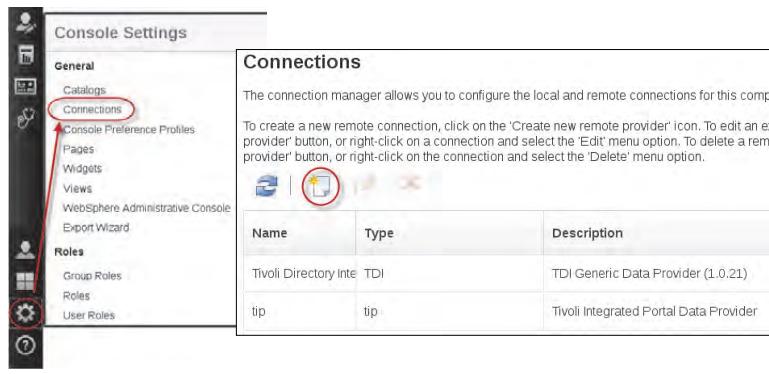
IBM Tivoli Monitoring dashboard data provider (continued)

There are several ways to enable the dashboard data provider. The graphic display shows a very simple way to see if the dashboard data provider is enabled.

During the installation, you can select Y to enable the dashboard data provider. You can also reconfigure the Tivoli Enterprise Portal Server service to select an option to enable the dashboard data provider.

Creating connections

- Use the Connections management widget in the Console Settings folder to manage existing connections in the console workspace
- Click the Create new remote provider icon



Integrating IBM Tivoli Monitoring and DASH

17

© Copyright IBM Corporation 2016

Creating connections

Click the gear icon. From Console Settings, click **Connections**. Click the icon that looks like a sheet of paper with a star-burst icon over the paper. This action opens a new connection to be created.

Creating connections (continued)

- Enter the following information:
 - Host name: Fully qualified host name where the portal server is running
 - Port: 15200 for HTTP, 15201 for HTTPS
 - Path: /ibm/tivoli/rest
 - Name and password: Valid user for IBM Tivoli Monitoring
- Click Search

Server information

* Protocol: HTTP * Host name: sysitmsles.poc.ibm.com * Port: 15200

* Path: /ibm/tivoli/rest

Connection goes through a firewall

Firewall address Firewall port

Use the following credentials to query the remote data providers

* Name: sysadmin * Password: *****
* Confirm password: *****

Search

Integrating IBM Tivoli Monitoring and DASH

18

© Copyright IBM Corporation 2016

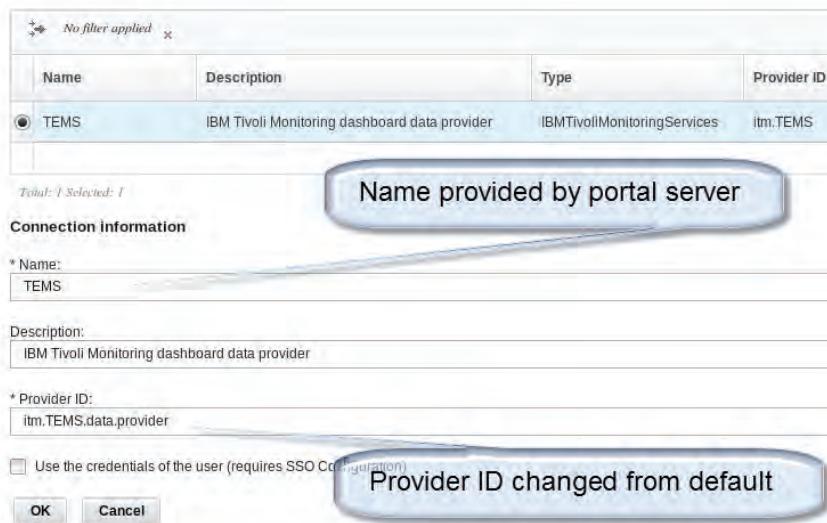
Creating connections (continued)

The connection information for IBM Tivoli Monitoring that we are using in this course are:

- Protocol HTTP. The other choice is HTTPS
- Host name: sysitmsles.poc.ibm.com is the host for the IBM Tivoli Monitoring portal server
- Port: 15200 is used to access the portal server client information
- Name: sysadmin is the user ID to log in to the portal server. It could be any other ID defined to IBM Tivoli Monitoring but that ID might require authorizations to use different functions.
- Password: object00 is the password for sysadmin. If you use a different user ID, you need to provide the password for that user ID.

After this information is provided, click Search to locate the connection information.

Selecting a connection



Selecting a connection



Note: You have to change the Provider ID to ITMSD only if you install the IBM Infrastructure Management Dashboards for Servers component.

The information for the TEMS name displays the description, type, and provider ID. The description is IBM Tivoli Monitoring dashboard data provider.

The provider ID is changed in your labs to ibm.TEMS.data.provider.

Click **OK** to save the connection information.

Verifying that the dashboard data provider is enabled

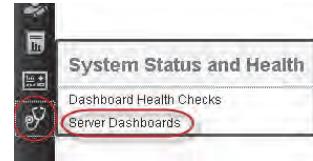
- Check the Tivoli Enterprise Portal Server configuration to verify that the dashboard data provider is enabled
- Use Manage Tivoli Enterprise Monitoring Services or the command line
- Start Manage Tivoli Enterprise Monitoring Services on the computer where the portal server is installed
- Right-click Tivoli Enterprise Portal Server
- Click Reconfigure (Windows) or click Configure (Linux)
 - The Common Event Console Configuration window opens
- Click OK to accept the current values
- On the Configure Tivoli Enterprise Portal window, select the Dashboard data provider tab

Verifying that the dashboard data provider is enabled

These are ways to verify that the dashboard data provider is enabled. If the dashboard data provider is not enabled, follow the information on this slide to reconfigure and enable the dashboard data provider.

Verifying Tivoli Monitoring dashboard

- Click the stethoscope icon
- Select Server Dashboards
- Compare the Managed System Groups and situation severities with the portal client presentation of these entries

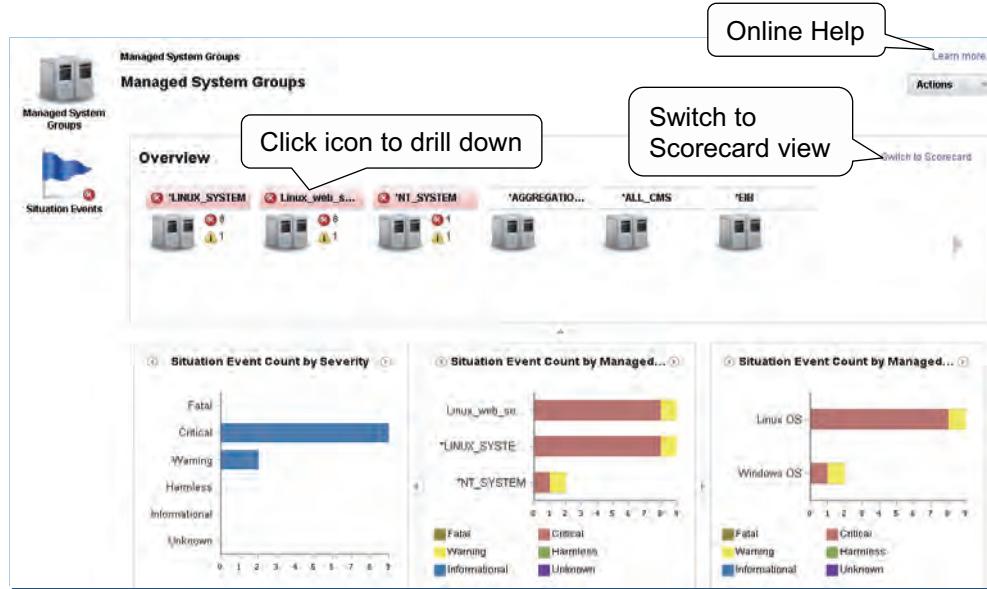


A screenshot of the 'Server Dashboards' interface. The title bar says 'Server Dashboards'. On the left, there are two sections: 'Managed System Groups' (with an icon of two server racks) and 'Situation Events' (with an icon of a flag). In the center, there is a 'Managed System Groups' section with the same two icons. Below it is an 'Overview' section with five server rack icons. Each icon has a status indicator: the first two are red with '1' (error), the third is yellow with '3' (warning), and the last two are green with '1' (information). The bottom of the screen shows a footer with 'Integrating IBM Tivoli Monitoring and DASH' on the left, '21' in the center, and '© Copyright IBM Corporation 2016' on the right.

Verifying Tivoli Monitoring dashboard

If you have the System Status and Health installed, by clicking Server Dashboards, you see the Managed System Groups and the Overview. Click one of the groups to see more information about the members of each group.

Carousel view



Integrating IBM Tivoli Monitoring and DASH

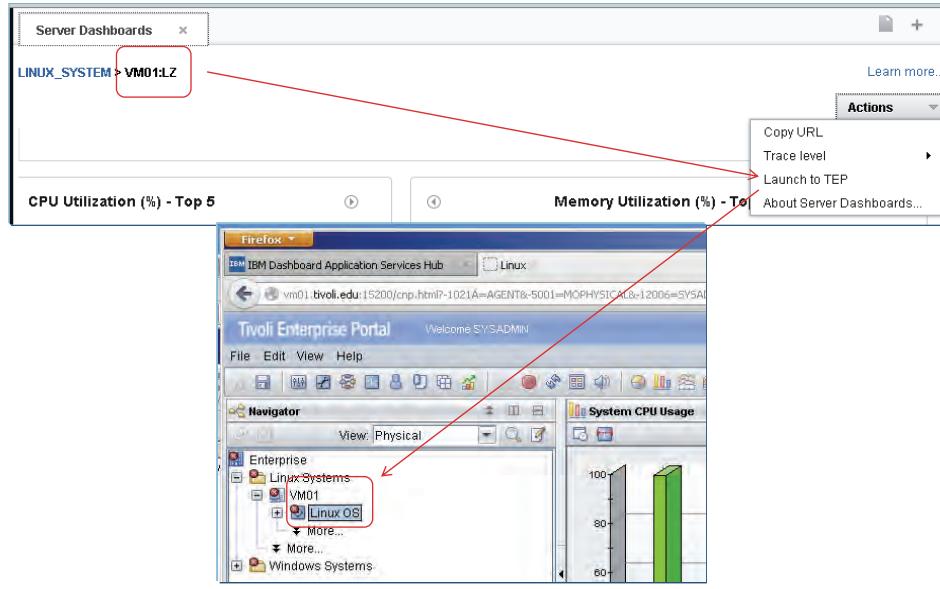
22

© Copyright IBM Corporation 2016

Carousel view

These widgets are product provided and can not be altered.

Starting the Tivoli Enterprise Portal client



Starting the Tivoli Enterprise Portal client

You can launch the Tivoli Enterprise Portal client in context from the Actions area.

Lesson 3 Building dashboards

IBM Training



Lesson 3 Building dashboards

24

© Copyright IBM Corporation 2016

Editing a widget and specifying data sets

- Data providers partition their data into one or more data sources and for each data source, there can be one or more dataset.
- When you edit a widget, you select the dataset that you want to retrieve data from.
- The ITM dashboard data provider has a data source for each agent type that has application support installed on the TEPS. The datasets for a data source correspond to an attribute group of a monitoring agent.
 - For example the Linux OS agent has datasets such as Linux Disk, Linux CPU, Linux Process, Linux Network, etc.
- When choosing a dataset, you are selecting the attribute group that contains the metrics you want to display in a widget.

Editing a widget and specifying data sets

Many data sources are provided with the product. They contain even more data sets within the data sources.

Examples of ITM Data Providers and Datasources

The screenshot displays the TEMs interface with the following navigation path:

- Provider: TEMS > Datasource: IMS**
- Local CF IMS DS OSAM**
- OSAM Subpools**
- VSAM/OSAM Databases**
- Provider: TEMS > Datasource: Internet Services**
- KIS HOST STATISTICS**
- KIS_HOST_STATISTICS**
- Provider: TEMS > Datasource: J2SE Application**
- JDK - Operation System**
- JDK_Operation_System**
- Provider: TEMS > Datasource: Linux KVM Agent**
- KV1 HOST CPU**
- KVL_HOST_CPU**
- KV1 HOST MEMORY**
- KVL_HOST_MEMORY**
- KV1 HOSTS**
- KVL_HOSTS**
- Provider: TEMS > Datasource: Linux OS**
- Linux Host Availability**
- Linux_Host_Availability**
- Linux OS Config**
- Linux_Os_Config**
- Datasets Found: 240**
- Provider: TEMS > Datasource: Linux OS**
- Linux CPU**
- Linux_CPU**
- Linux CPU (Superseded)**
- Linux CPU Available**
- Linux_Cpu_Available**
- Linux CPU Available (Superseded)**
- Linux CPU Configuration**
- Linux_Cpu_Config**
- Linux Disk**
- Linux_Disk**
- Linux Disk (Superseded)**
- Linux Disk IO**
- Linux_Disk_Io**
- Linux Disk IO (Superseded)**
- Linux Disk Usage Trends**
- Linux_Disk_Usage_Trends**
- Datasets Found: 48**
- Linux Disk Usage Trends**
- Linux_Disk_Usage_Trends**
- Datasets Found: 48**
- Linux File Comparison**
- Linux_File_Comparison**
- Linux File Information**
- Linux_File_Information**
- Linux File Pattern**
- Linux_File_Pattern**
- Linux Group**
- Linux_Group**
- Linux Host Availability**
- Linux_Host_Availability**
- Linux IO Ext**
- KLZ_IO_Ext**
- Linux IO Ext (Superseded)**
- Linux IP Address**
- Linux_IP_Address**
- Linux LPAR**
- KLZ_LPAR**
- Linux Machine Information**
- Linux_Machine_Information**
- Linux Network**
- Linux_Network**
- Linux Disk Usage Trends**
- Linux_Disk_Usage_Trends**
- Datasets Found: 48**
- Linux Network (Superseded)**
- Linux NFS Statistics**
- KLZ_NFS_Statistics**
- Linux NFS Statistics (Superseded)**
- Linux Network**
- Linux_Network**
- Linux Sockets Detail (Superseded)**
- Linux_Sockets_Detail**
- Linux Sockets Status**
- KLZ_Sockets_Status**
- Linux Sockets Status (Superseded)**
- Linux_Sockets_Status**
- Linux Swap Rate**
- KLZ_Swap_Rate**
- Linux Swap Rate (Superseded)**
- Linux_Swap_Rate**
- Linux OS Config**
- Linux_Os_Config**
- Linux Process**
- KLZ_Process**
- Linux Process (Superseded)**
- Linux_Process**
- Linux Process User Info**
- KLZ_Process_User_Info**
- Linux Process User Info (Superseded)**
- Linux_Process_User_Info**
- Linux RPC Statistics**
- KLZ_RPC_Statistics**
- Linux RPC Statistics (Superseded)**
- Linux_RPC_Statistics**
- Linux Sockets Detail**
- KLZ_Sockets_Detail**
- Linux Sockets Detail (Superseded)**
- Linux_Sockets_Detail**
- Datasets Found: 46**
- Datasets Found: 46**

Examples of ITM Data Providers and Data Sources

This slide shows just an extraction of some of data sources. One data source is expanded to show all of the data sets within that data source. Each installed agent has one or more data sources and many datasets provided by the product.

Creating a dashboard page

- Three methods to open the dashboard workspace:
- Click the **Pages** widget in the **Console Settings** menu, and click **New Page**

The screenshot shows the 'Console Settings' interface. On the left, under 'General' settings, the 'Pages' option is highlighted with a red circle. A red arrow points from this 'Pages' link to a larger screenshot of the 'Pages' configuration screen. In this configuration screen, the 'New Page...' button is circled in red. Below it, there's a 'Select' dropdown with an arrow pointing to it, and a list of options: 'Name', 'Default', and 'Administrative'. To the right of the configuration screen, four numbered steps are listed:

2. Click the plus symbol in the upper right of the console
3. Position the cursor over the arrow in the console Welcome page
4. Click **build a page**

Below these steps is another screenshot of the 'Get started' welcome page. On the right side of this page, there are three buttons: 'create a widget', 'build a page' (which is also circled in red), and 'manage roles'.

Integrating IBM Tivoli Monitoring and DASH

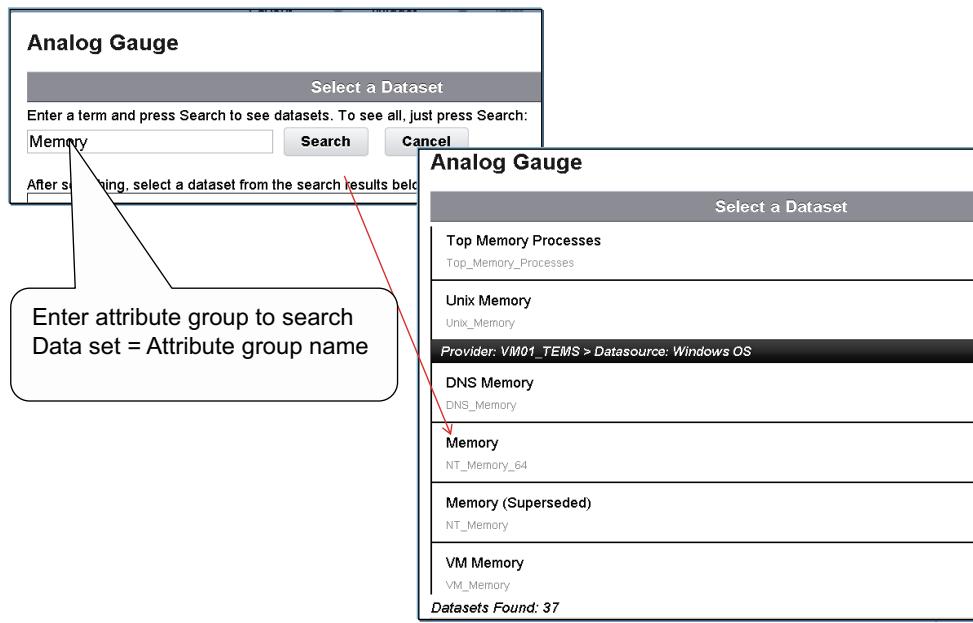
27

© Copyright IBM Corporation 2016

Creating a dashboard page

These are the three ways that you can begin to create a new dashboard page.

Selecting the data set



Integrating IBM Tivoli Monitoring and DASH

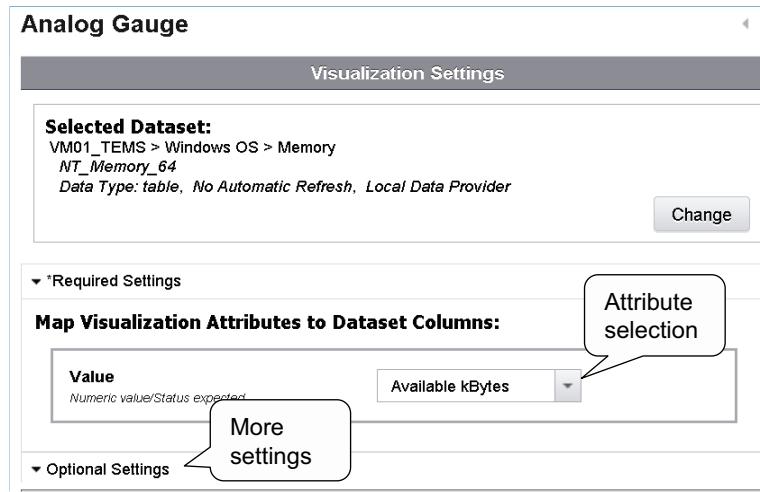
28

© Copyright IBM Corporation 2016

Selecting the data set

For your new page, you drag and drop widgets. This example has had an analog gauge dragged to the page editing area. The search was for Memory. Many datasets are displayed that you can select.

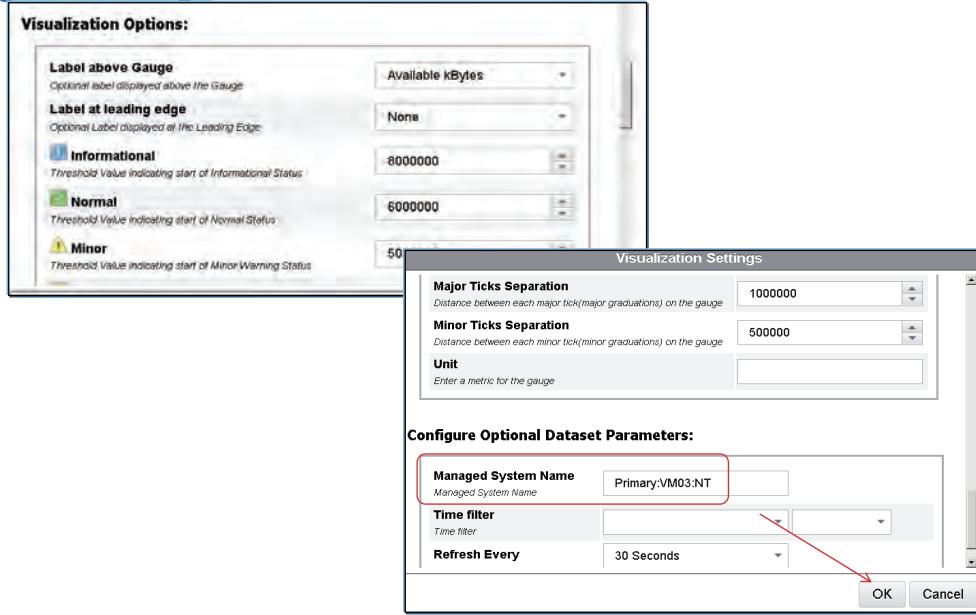
Selecting the attribute



Selecting the attribute

Memory was selected. The value is provided as the attribute **Available kbytes**.

Finishing the settings



Integrating IBM Tivoli Monitoring and DASH

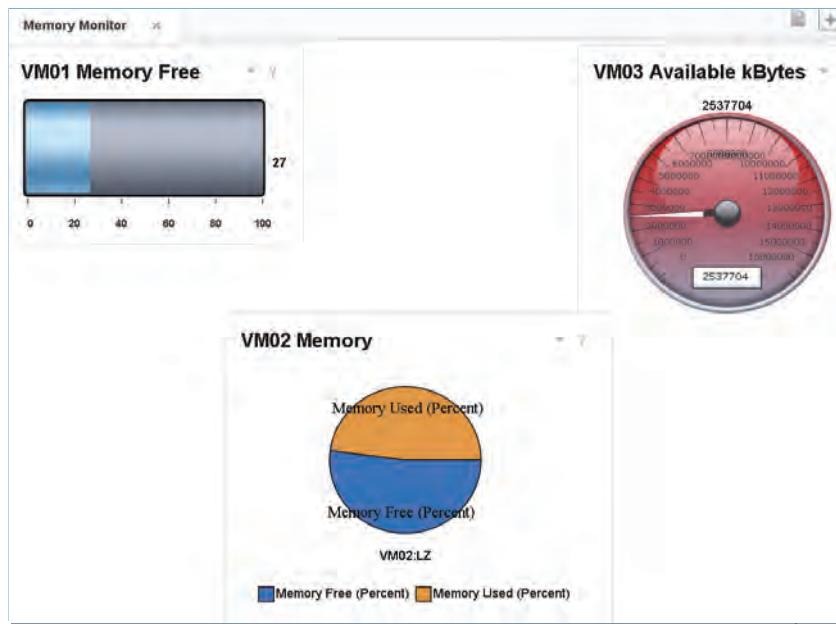
30

© Copyright IBM Corporation 2016

Finishing the settings

There are some optional settings that you can enter. On the slide, Primary:VM03:NT is a single Managed System that is selected to display that system's monitoring information. You can enter a group name as the managed system name.

Memory Monitor dashboard



Integrating IBM Tivoli Monitoring and DASH

31

© Copyright IBM Corporation 2016

Memory Monitor dashboard

This is an example of using three different widgets to show similar memory information for three different managed systems. It was used as a trivial sample just to explain some basic information about widgets.

Lesson 4 Troubleshooting

IBM Training



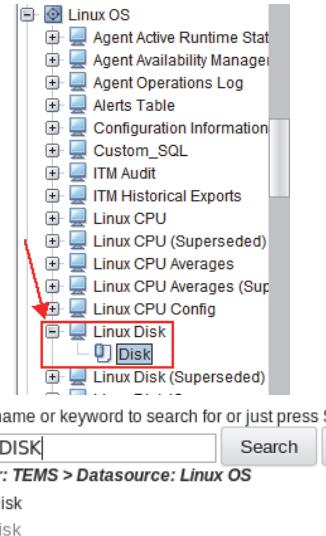
Lesson 4 Troubleshooting

32

© Copyright IBM Corporation 2016

Locating attributes and data sets

- The TEPS data provider communicates directly with the TEPS and TEMS to retrieve the dataset metrics.
- If there is a specific metric that you see inside the TEP but do not know where to find the data set you can obtain this information from the TEP.
 - Right-click workspace with the metric you want, “Click here to assign a query” to see the location of the data set metric
- Now the user can search for the dataset inside DASH when creating a widget.



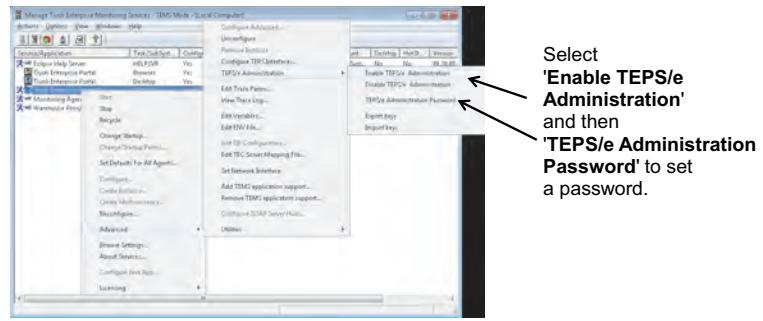
Locating attributes and data sets

You can look at the attribute groups and attributes to give you an idea of what kind of information is available in the datasets within data sources.

When Linux Disk is displayed, you can see Disk. Linux Disk corresponds to the Linux OS data source Linux Disk data set.

Data Provider logs and traces

- ITM DataProvider logs are available on the TEPS server in the eWAS directory:
- Windows : %CANDLE_HOME%\CNPSJ\profiles\ITMProfile\logs\ITMServer\SystemOut.log
- UNIX/Linux: \$ITM_HOME/<platform>/iw/profiles/ITMProfile/logs/ITMServer/SystemOut.log
- To enable verbose tracing, the eWAS console ISCLite must be enabled:



Integrating IBM Tivoli Monitoring and DASH

34

© Copyright IBM Corporation 2016

Data Provider logs and traces

This slide is an example to display log information.

Select the component (Tivoli Enterprise Portal Server) > Advanced > TEPS/e Administrator. From there, you can select additional functions.

DataProvider logs contd.

- Log in to ISClite with **wasadmin** user ID
 - <https://<TEPS machine>:15206.ibm/console>
- Select:
 - Troubleshooting
 - Logs and trace
 - ITMServer
 - Diagnostic Trace

The screenshot shows two pages from the WebSphere Integrated Solutions Console:

- Login Screen:** Shows a key icon and fields for User ID (wasadmin) and Password (****). A 'Log in' button is at the bottom.
- Logging and tracing Page:** Under 'Logs and trace', it lists resources: ITMServer, ITMNode, and nc042012. A red box highlights the 'Logs and trace' link under 'Troubleshooting' in the left sidebar, and another red box highlights the 'Logs and trace' link in the main content area.

Integrating IBM Tivoli Monitoring and DASH

35

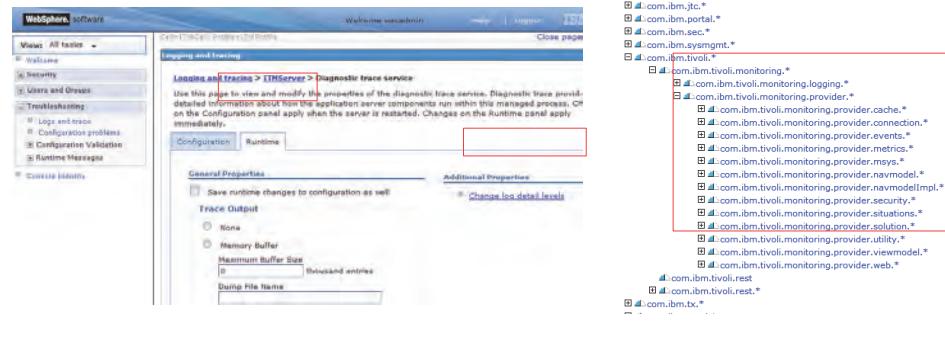
© Copyright IBM Corporation 2016

DataProvider logs contd.

Select **Troubleshooting > Logs and Traces > ITMServer > Diagnostic Trace**. This is continued on the next slide.

DataProvider logs contd.

- Select:
 - Runtime tab for the changes to be effective immediately
 - Change log detail levels
 - Expand [All Components]
- DataProvider code belongs to package com.ibm.tivoli.monitoring.provider



DataProvider logs contd.

Select Configuration and then select one of the files displayed in the example on the right side of the slide. You can examine the information from that log file.

Exercise 1

Exercises

Complete Unit 7 Exercise 1 in the exercise guide. This is the only exercise for the unit.

Instructor demonstration

IBM Monitoring 8.1.3 integrating with Dashboarding



Summary

Now that you have completed this unit, you can perform the following tasks:

- When you complete this unit, you can perform the following tasks:
- Describe the Tivoli Monitoring architecture
- Enable the Tivoli Monitoring data provider
- Explain how the dashboard data requester connects to IBM Tivoli Monitoring 6.3
- Use the Tivoli Monitoring data provider with widgets
- Create a custom dashboard

Unit 8 Integrating with Tivoli Business Service Manager

IBM Training



Unit 8 Integrating with Tivoli Business Service Manager

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this unit, you learn how to use the IBM Tivoli Business Service Manager UI data provider to serve business service data to dashboard pages. You learn how to configure different widgets with several of the available data sets, including topology data. You learn how to create multiple-page dashboards that show high-level data views and can show context-driven detail information in secondary dashboard pages. Finally, you learn how to configure right-click menu data to data sets. The right-click menus are available to compatible widget types, such as list, table, or tree table widgets.

Objectives

After completing this unit, you should be able to perform the following tasks:

- Describe the function and relationships of templates, rules, and services in Tivoli Business Service Manager
- Describe the Tivoli Business Service Manager primary architectural components
- Describe the Tivoli Business Service Manager data provider architecture
- Describe the general properties of the Tivoli Business Service Manager data sets
- Use Tivoli Business Service Manager data sets to show business service status and key performance indicators in business dashboards

Lesson 1 Tivoli Business Service Manager overview

IBM Training



Lesson 1 Tivoli Business Service Manager overview

Business service management definitions

- A **business service** is any combination of applications, middleware, security, storage, networks, and other key performance indicators that collectively support a higher-level business component
- **Business service management** is a method of grouping and tracking the status and business impact of the components that comprise a business service
- A **business service model** is any logical grouping of the components that comprise a business service

Business service management definitions

This slide shows several key concepts and terms that are used throughout this unit.

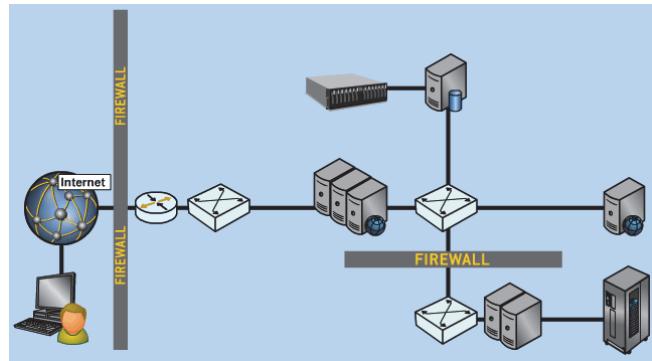
A business service is any logical grouping of information technology resources that support some higher level business function.

Business Service Management is the grouping of monitored resources supporting a higher-level logical view of a business service. Historically, system management monitors focused on a particular aspect of an enterprise information technology resource. Examples of these might be monitoring network components, disk usage, or computer system CPU performance.

Tivoli Business Service Manager uses the concept of a business service model as a logical way of combining related monitors that support a high level business service. Examples of typical business service models might be an online banking system, e-commerce, or credit card processing. By mapping the physical monitored resources with a logical business service, you are easily able to see the impact of the failure of a monitored resource on a business service.

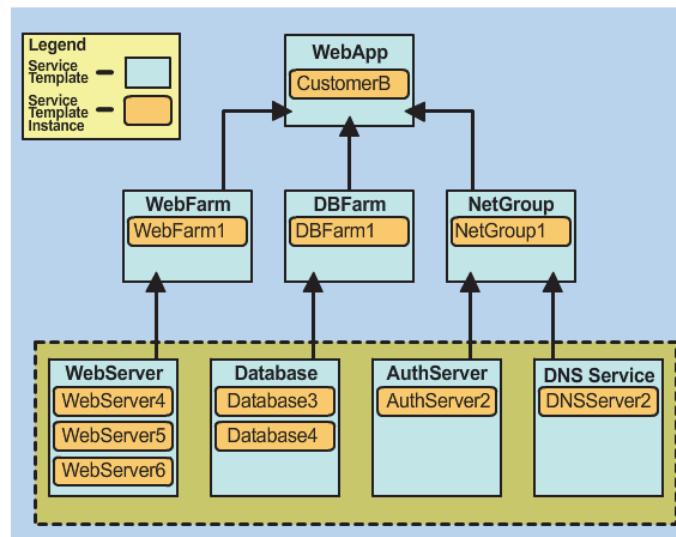
Business service example: Online banking

To reflect the status of a customer online banking application, the application and all supporting infrastructure must be tracked as one logical entity



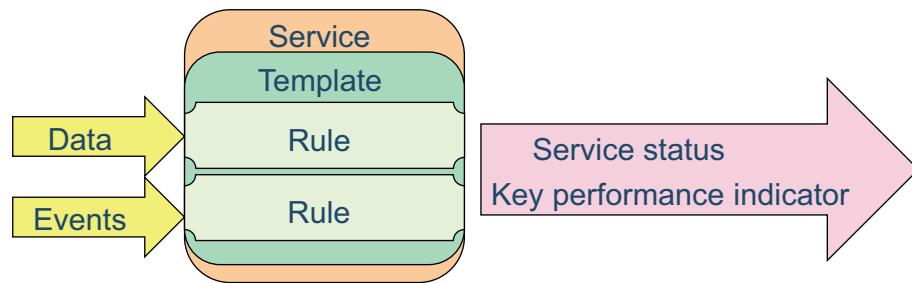
Customer experience	Application servers	Firewalls
Routers	Databases	Web servers
Database clusters	Web server farms	

Defining business services with service models



- Templates model general business service characteristics
- Services are created from templates

Service status and key performance indicators



- Service status or key performance indicators are tracked with one or more rules that are associated with a service template
- The types of rules are as follows:
 - Incoming status
 - Dependency
 - Numerical formula
 - Auto population
 - ESDA

Service template

- Templates model business processes and business process behavior
- Templates can correspond to physical resources or logical processes within a business
 - Physical resources
 - Web servers
 - Databases
 - Logical processes
 - Web applications
 - Supply chain performance



Business services and service status



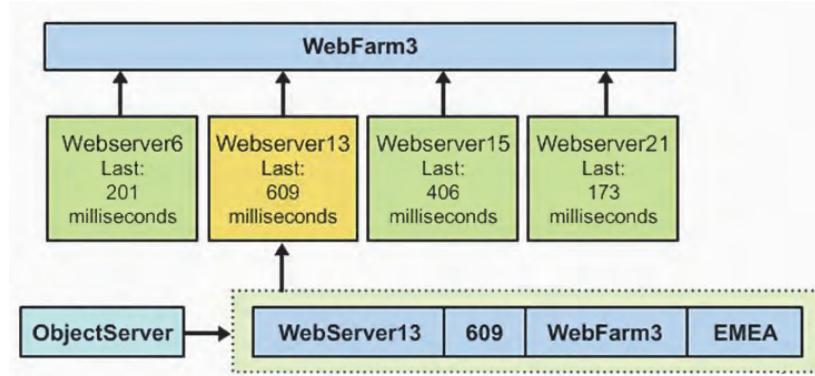
Tivoli Business Service Manager provides two major functions:

- Model business services
 - Manually
 - Automatically
- Track business service status and key performance indicators
 - Status (Good, Marginal, Bad, Unknown)
 - Monitored events
 - Business data

Incoming status rules

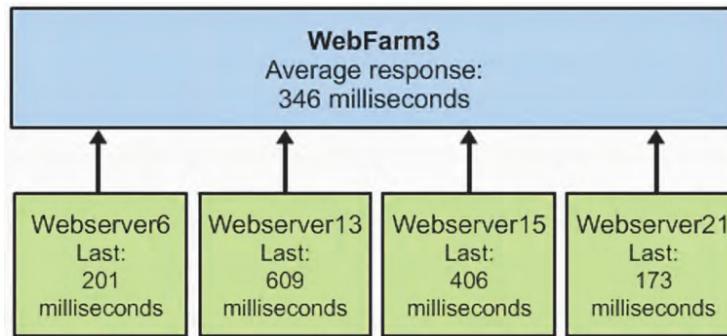
- Evaluate the status of a business service or calculate a key performance indicator
- Evaluate monitor events or business data

Example: Assign a **Marginal** status to any web server service with a response time that is greater than 500 milliseconds



Dependency rules

- You create hierarchical service models by configuring parent-child dependency relationships between services
 - The relationships are defined with *dependency rules*
- Example: Calculate the average response time of a web farm with the response times of all child web server instances

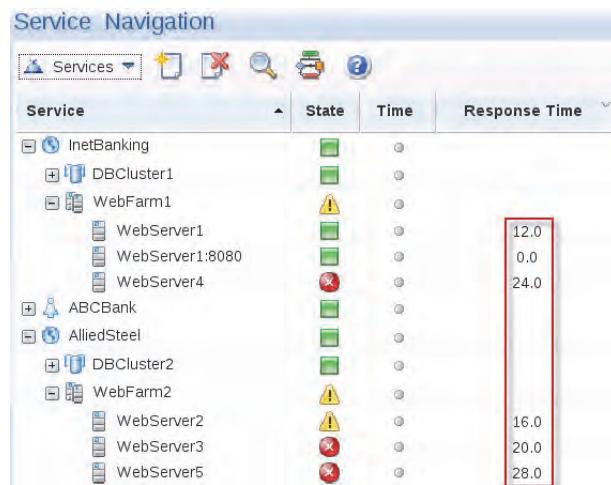


Numerical modeling

- You can associate performance measurements with a service
- You can use numerical calculations as metrics in determining a service status
- You can use calculated metrics as a key performance indicator (KPI) in a business dashboard

Example: Calculate the response time for a web server service model with a monitor event field value

Showing key performance indicators



You can view KPIs with service status and SLA data

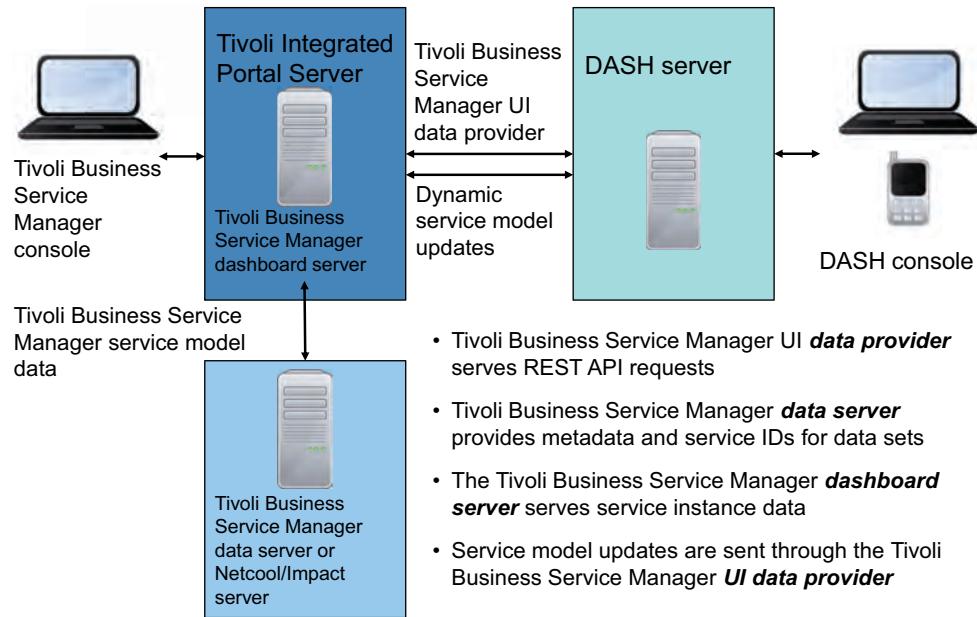
Lesson 2 Tivoli Business Service Manager data provider architecture

IBM Training



Lesson 2 Tivoli Business Service Manager data provider architecture

Tivoli Business Service Manager data provider architecture diagram



UI data provider overview

- The Tivoli Business Service Manager UI data provider displays the following service model attributes:
 - Status
 - Metrics or KPIs
 - Topology
- The UI data provider is connected to the Tivoli Business Service Manager dashboard server, and the service model data is accessed through this connection

Data sources

- The data sources are in the following two categories:
 - Collections without relationships: Flat, or tabular, data sets
 - Collections with relationships: Topology data sets
- The data provider has five data sources:
 - Services
 - Templates
 - Primary templates
 - Topology
 - Tree templates

Data sources

Flat collections of services with a view like a spreadsheet. Each row represents a service and the columns represents the data available for all of the collected services.

Collections of services along with parent and child relationships. These relationships correlate to the relationships you see in the Tivoli Business Service Manager service viewer.

- TBSM_Services—single flat, large model collection of all services with default properties
- TBSM_Templates—single flat, large model collection per TBSM template
- TBSM_PrimaryTemplates—single flat, large model collection per template where services all have template as the “primary tag”
- TBSM_Topology—single topology dataset based on origin services parameter

All collections have “nodes” for each service in the collection.

Nodes have properties of various types.

TBSM supports string, double, and status types.

Topology collection also has relationships.

Relationships can have properties –none available for TBSM topology.

TBSM Services data source

- Includes basic service data, for scalar or collection widgets
- Does not include template metrics (KPIs)
- Includes all service instances
- Has the following base properties, which are common to all datasets:
 - Service instance name
 - Display name
 - Description
 - Primary template
 - Tivoli Business Service Manager status
- Has the following parameters:
 - Tivoli Business Service Manager service
 - Gather contained services

TBSM Templates data source

- Is based on all defined service model templates
- Includes all services that are tagged with the template, either as a primary or secondary service
- Has base properties plus the following ones:
 - Numeric rule properties, excluding ESDA and automatic population rules
 - Status property for each rule that defines status thresholds
 - Properties for all user-defined attributes
- Has the following parameters:
 - Tivoli Business Service Manager service
 - Gather contained services

TBSM Templates data source

Templates may be marked as not available as dataset:

Set additional attribute “TBSM_DataProviderCollection=false”

Default is true when attribute does not exist (default)

TBSM Primary Templates data source

- Has template data from primary template and any assigned secondary templates
- Includes member services, which are all services with the selected template assigned as the primary template
- Has base properties plus the following ones:
 - Numeric rule properties, excluding ESDA and automatic population rules
 - Status property for each rule that defines status thresholds
 - Properties for all user-defined attributes
- Has the following parameters:
 - Tivoli Business Service Manager service
 - Gather contained services

TBSM Primary Templates data source

Many properties may be null when templates are found that are not tagged to all services in the dataset

Templates may be marked as not available as dataset, just as with basic template datasets

TBSM Topology data source

- Is a collection of basic service data plus relationships for the topology and tree table widgets
Template metrics and KPIs are not available
- Includes member services from the selected origin service and all parent and child services in the dependency
- Includes relationship data for use with topology and tree table widgets
- Has base properties
- Has the following parameters:
 - Tivoli Business Service Manager service
 - Topology levels down
 - Topology levels up

TBSM Topology data source

Relationships do not have any properties added by Tivoli Business Service Manager

TBSM Tree Template data source

- Tree templates are created to filter service model data in a service tree portlet in the Tivoli Integrated Portal server
- Tree templates can be used in DASH widgets to filter the services that are included in the data set, including topology data
- Includes relationship data for use with topology and tree table widgets
- Has base properties
- Has the following parameters:
 - Tivoli Business Service Manager service
 - Topology levels down
 - Topology levels up

DayTrader Revenue				
Service	CLIENTLOST	CLIENTTOTAL	VIPLOST	VIPTOTAL
Chicago	2.3	2.4	0.6	2.5
Dallas	0.5	3.8	1.1	3.0
Los Angeles	3.7	4.3	0.6	2.3
New York	2.1	3.2	0.0	1.4
Seattle	3.8	2.1	1.9	0.6

Dynamic updating

Tivoli Business Service Manager data sets support dynamic updating

- Dynamic updating uses the same listener interfaces that are defined for the client (Tivoli Integrated Portal console) model
- Updates are sent for the following conditions:
 - Service rule value changes, including status
 - Service status changes
 - Service or service dependencies, which are created, deleted, or modified
 - Template rules and attributes that are added or removed, which affects the list of available data sets
- The REST layer queues the update notifications to the client
 - Notifications are pushed to the widget client
 - A minimal delay is built into the queue polling and delivery to the widget

Dynamic updating (continued)

- Dynamic updating includes refresh notification
 - Causes another request for the full dataset
 - Depends on the complexity of determining atomic updates
 - Example: Removing a relationship from a topology data set might force rebuilding
 - Updates that affect sorting and filtering for a large model will schedule a refresh notification, with a 30-second default interval
- Dynamic updating includes creation and deletion of templates, and changes to attributes
 - List of provider data sets typically update immediately
 - Addition and removal of a service property requires closing and opening a page
 - The page configuration might require an update to remove or add an attribute
 - This type of activity typically occurs in a development environment rather than production

Context menus

- Tivoli Business Service Manager contributes context menu items that are merged with the default widget menus
 - Only available for service data in table, tree table, and topology widgets
- Menu items are defined in **DatasetConfig.xml**
 - Stored in the Tivoli Business Service Manager database as a treemap template artifact
 - Maintained with **getartifact** and **putartifact** commands
- Default context menu items are as follows:
 - Clicking **Show Business Impact** opens the Service Availability page with the context of the selected service
 - Menu items are added as an **action** subcategory of **menuactions**

```
putArtifact -U xxxx -P xxxx -c menuactions -s action -n \path\NewAction.xml
```

Lesson 3 Using data widgets with Tivoli Business Service Manager data sets

IBM Training

IBM

Lesson 3 Using data widgets with Tivoli Business Service Manager data sets

Correlating template rules and template data sets

The screenshot illustrates the correlation between template rules in the Tivoli Integrated Portal console and dataset properties in the DASH console.

Template rule in Tivoli Integrated Portal console:

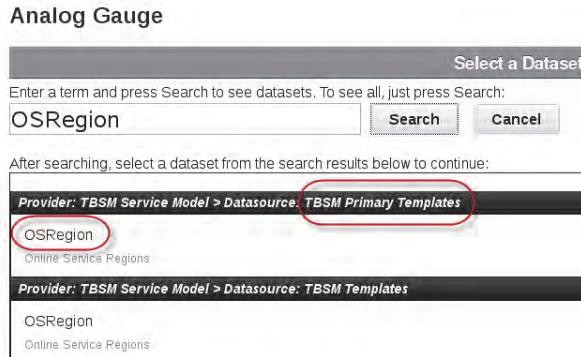
- Rules tab selected.
- Rule "RegSumHigh" is highlighted with a red box and has a red arrow pointing up to it from the DASH console.
- Rule "RegSumCritical" is also highlighted with a red box.
- Other rules listed include: CriticalDBTickets_Sum_OSDBFarm_to_OSRegion, CriticalNetworkTickets_Sum_OSNetwork_to_OSRegion, HighNetworkTickets_Sum_OSNetwork_to_OSRegion, CriticalWebTickets_Sum_OSWebfarm_to_OSRegion, HighWebTickets_Sum_OSWebfarm_to_OSRegion, and TotalDBTickets_Sum_OSDBFarm_to_OSRegion.

Data set properties in DASH console:

- Map Visualization Attributes to Dataset Columns dialog box.
- Value dropdown menu shows "None" and "Numeric value/Status expected".
- Optional Settings dropdown menu shows "None" and several dataset columns:
 - CriticalDBFarmTickets_Sum_OSDBFarm_to_OSRegion
 - CriticalNetworkTickets_Sum_OSNetwork_to_OSRegion
 - CriticalWebFarmTickets_Sum_OSWebFarm_to_OSRegion
 - DBFarmStatus
 - HighDBFarmTickets_Sum_OSDBFarm_to_OSRegion
 - HighNetworkTickets_Sum_OSNetwork_to_OSRegion
 - HighWebFarm_Tickets_Sum_OSWebFarm_to_OSRegion
 - NetworkStatus
 - RegSumCritical
 - RegSumHigh
 - RegionTicketsSum
 - RegionTicketsSum (Status)

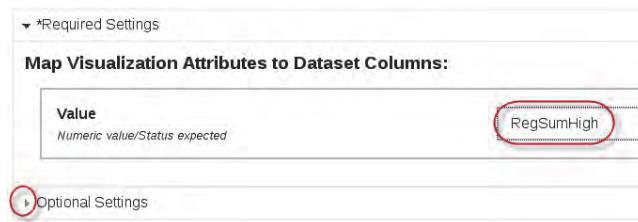
Example: Analog gauge with TBSM Primary Templates data source

In this example, selecting OSRegion from the **TBSM Primary Templates** data source returns all services that are tagged with **OSRegion** as the primary template



Selecting the analog gauge data set column value

- The analog gauge is a scalar widget
- Only one dataset column must be selected for the value
Several optional settings are available
- In this example, the **RegSumHigh** rule value is selected for the analog gauge value



Optional visualization settings: Labels

Labels above and to the left of the gauge can show service properties, such as the service display name, or rule KPI

Visualization Settings

Title: High Ticket Count

Visualization Options:

- Display as Half-moon Gauge**: Optional, select to display the gauge with half height.
- Label above Gauge**: Optional, select a property from the list to show its value or type in any custom label.
- Label at leading edge**: Optional, select a property from the list to show its value or type in any custom label.
- Label at center of Gauge**: Optional, select a property from the list to show its value or type in any custom label.
- Font Size**: Optional, select font size for the labels.
- Font Family**: Optional, select from the list or type in a valid font family name for the labels.
- Font Color**: Optional, click on the button to choose a font color for the labels.

The "Label above Gauge" section is highlighted with a red box. To its right is a circular gauge with a value of 39. The word "Asia" is written above the gauge, and a red arrow points from the "Label above Gauge" section to the word "Asia". A green checkmark is located in the bottom right corner of the gauge area.

Optional visualization settings: Values

- Assign threshold values
 - The values can be different from template thresholds
 - They are not automatically configured with service model template values

Visualization Settings

Messages: 0

Minimum Value: 0

Maximum Value: 100 (circled)

Informational: 0

Normal: 40 (circled)

Minor: 60 (circled)

Major: 80

Critical: 100

Fatal: 120

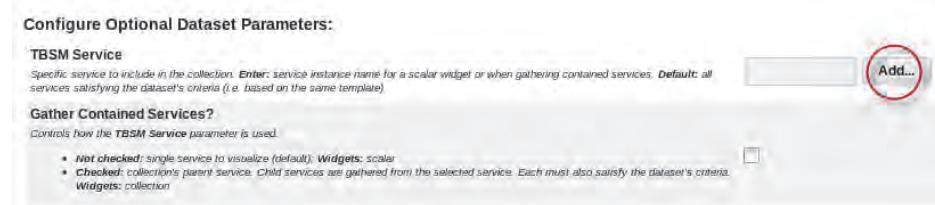
Show Threshold Strip:



Optional visualization settings: Values

Analog gauge: Selecting a service

- If left blank, the service value is provided by the context from a source event wire or the first matching service in the data set
- To specify a service, enter one or more service names directly, separated by spaces
 - Alternatively, click Add to select from a list of known services



Analog gauge: Selecting a service

Service name strings can be very long, and seem almost random. The display name is easier to read.

Clicking the Add button brings up a list of all of the services in the TBSM environment. Clicking a service will insert that name into the TBSM Service area of the widget.

Analog Gauge: Enhanced Parameter Selector

- Select one or more services at a time
 - Helps to avoid typographical errors

Enhanced Parameter Selector - TBSM Service

Select the desired value(s) for this parameter and click Ok or Apply to add the additional value(s). Use the table's sort and/or filtering to assist in locating values.

Display Name	Description	TBSM Status	Service Instance Name	Primary Tag
ABCBank	Auto-created from R	Marginal	ABCBank	Online_S
AIX		Good	SCR_Servers_Unix	SCR_Top
All		Good	SCR_Servers_ALL	SCR_Top

Configure Optional Dataset Parameters:

TBSM Service

Specific service to include in the collection. Enter: service instance name for a scalar widget or when gathering contained services. Default: all services satisfying the dataset's criteria (i.e. based on the same template)

AnyBank ABCBank

Gather Contained Services?

Controls how the TBSM Service parameter is used.

- Not checked: single service to visualize (default); Widgets: scalar
- Checked: collection's parent service. Child services are gathered from the selected service. Each must also satisfy the dataset's criteria.

Widgets: collection

Analog gauge: Enhanced parameter selector

All of the defined services appear after the Add button was clicked. Each time you click an entry from this list, it will insert this into the service field. You can't filter the list and might have to scroll through a lot of names, but it is more accurate than typing the service name.

You can add multiple service names in the field. When you move the cursor away from the service names, it appears in a green color.

Example: Tree table and topology data source

- You must select the topology or tree template data source for the tree table or topology widget
- No visualization attribute configuration is required

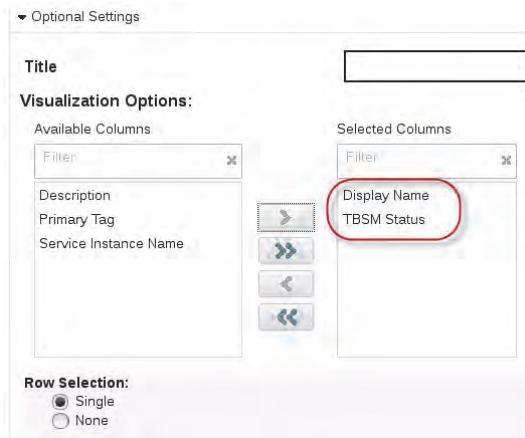
The screenshot shows the 'Tree Table' configuration dialog. On the left, there's a search bar with 'Topology' typed in, a 'Search' button, and a 'Cancel' button. Below the search bar is a message: 'Enter a term and press Search to see datasets. To see all, just press Enter.' A 'Select a Dataset' button is at the top right. The main area is titled 'Tree Table'. It shows a search result for 'Topology' under 'Provider: TBSM Service Model > Datasource: TBSM Topology'. A 'TBSM Topology' dataset is selected, described as 'Collection of basic service data plus relationships for the Topology and Tree Table widgets'. On the right, there's a 'Selected Dataset:' section with details: 'TBSM Service Model > TBSM Topology > TBSM Topology', 'Collection of basic service data plus relationships for the Topology and Tree Table widgets', and 'Data Type: table, No Automatic Refresh, Remote Data Provider'. Below this are sections for 'Required Settings' (with a note: 'No visualization attribute found for mapping to dataset columns.') and 'Optional Settings'. At the bottom, there are copyright notices: 'Integrating with Tivoli Business Service Manager', '34', and '© Copyright IBM Corporation 2016'.

Example: Tree table and topology data source

We are working with topology data. A tree table widget breaks out into a tree.

Tree table widget: Optional settings

Select one or more columns from the data set

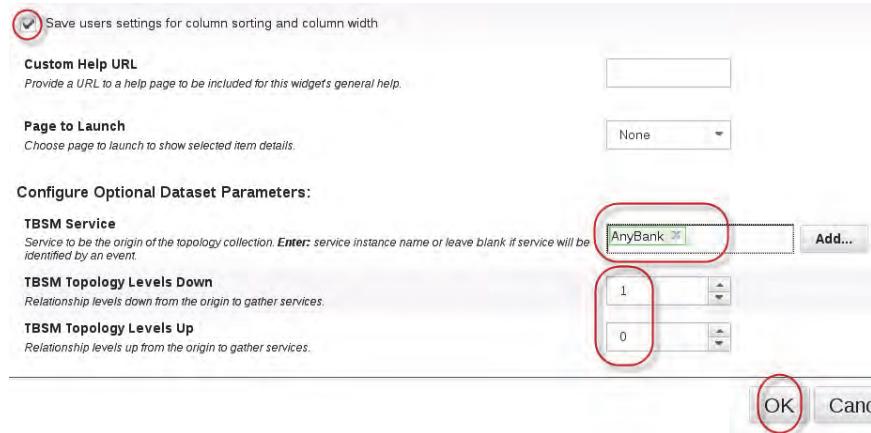


Tree table widget: Optional settings

We are working with the topology data. A tree table widget breaks out into a tree.

Optional settings: Save user settings

- Select option to save user sort and column width changes
- Select the root node for the tree table and the number of levels up and down to be shown



Optional settings: Save user settings

The service name is what you want to be at the primary point of the tree. Then you can select how many levels up or down the tree that the widget will display.

Completed tree table widget

- Tree table widget supports node information and rich hover view with OSLC registry services data
- The widget supports right-click context menus

The screenshot shows a 'Tree Table' interface. On the left is a tree view with nodes: AnyBank (Bad), East (Bad), Midwest, North, South, and West (Bad). The 'East' node is selected, highlighted with a dashed blue border. A tooltip appears over the 'East' node, displaying detailed information: Display Name: East, TBSM Status: Bad, Service Instance Name: EastAnyBank, Description: Auto-created from RegionalTickets at Tue Apr 15 23:34:16 UTC 2014 for Create OSRegion From OSWebFarm, and Primary Tag: OSRegion. At the bottom right, a context menu is open for the 'North' node, showing options: Show Business Impact (Tivoli Service Business Manager) and Properties... The menu item for 'North' is also highlighted with a dashed blue border.

Completed tree table widget

This shows a completed tree table widget. If you hover the mouse over an entry, it will show additional information or the general properties about that service.

Lesson 4 Troubleshooting

IBM Training



Lesson 4 Troubleshooting

Troubleshooting: Is it TBSM?

- UI data provider provides
 1. service model data and
 2. dynamic updates
- Participates in advanced widget capabilities of table, tree table, topology
 1. Tooltip content
 2. Properties data
 3. Context menu content

Some provided by TBSM

Some provided by widget

Table

Display Name	ClientLost	ClientTotal	VIPLost	VIPTotal
Chicago	2.5	3.1	2	2.8
Dallas	3.5	2.5	0.1	3.1
Los Angeles			0	
New York			2	
Seattle			3.1	

Dallas

Display Name:

TBSM Status:

Service Instance Name:

Description:

Primary Tag:

Troubleshooting: Is it TBSM?

Are you requesting the right information from TBSM? Hold the mouse over an entry and find out if you trying to show the correct service. The example shows the primary tag is the AEDayTrader.

Verify the DASH version

- Select the **About** menu by clicking the **Page Actions** icon in the upper right of the console
- In some cases, expected features may not be available with the currently installed version
 - Use the version information when creating a PMR



The screenshot shows the 'About' page of the DASH console. The URL in the address bar is <https://dash151.poc.ibm.com:16311/ibm/console/html/isclite/AboutPage.jsp>. On the left, a context menu is open with options: Open Page in New Tab, Add to My Startup Pages, Edit Page ..., Personalize Page, and About. The 'About' option is circled in red. The main content area is titled 'Console' and displays the following information:

System Detail	Value
Dashboard Application Services Hub	3.1.2.1
OS Name	Linux
OS Architecture	amd64
Java Version	1.6.0 - IBM Corporation
Browser Information	Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20100101 Firefox/24.0
Dojo Version	1.10.4-20150510-IBM (f4fef70)
Dashboard Widgets Version	3.2.2.0(build:irm-20Aug2015-irm_c3)

Verify the DASH version

Selecting About provides detailed information about the server itself. The version with fix pack numbers is important to the help center.

Incorrect service name

- Service names, especially services created from discovery data, include long random strings of data
 - Typographical errors can be introduced when manually entering the service name
- In the widget quick-edit window, in the **Configure Optional Dataset Parameters** section, you can click **Add** next to the **TBSM Service** field
 - You see a list of all available service names
 - You can select multiple services

Enhanced Parameter Selector - TBSM Service

Configure Optional Dataset Parameters:

TBSM Service
Service to be the tree root node. Enter: service instance name. Only leave blank if service will be identified by an event.

Service Instance Name	Primary Tag
Chicago	AEDayTrader
CICS	SCR_BSM_CICS
CICS Files	SCR_ZCICSFiles

Buttons: Add... (circled), OK (circled), Apply, Cancel

Bottom Buttons: OK (circled), Apply, Cancel

Bottom Left: Seattle X, Chicago X, New York X, Dallas X

Bottom Right: © Copyright IBM Corporation 2016

Incorrect service name

The display name can be different from the TBSM service instance name. The same display name can be used with multiple service names. But the service names must be unique.

Known problems and limitations

- The topology widget and tree table widgets support only the TBSM topology and tree template datasets
 - Tree table and topology require origin nodes
 - If flat dataset is used,
 - Topology widget displays error, unable to use the type of dataset specified (can't support large model datasets)
 - Tree table widget displays nothing, with no error, due to absence of "origin nodes"
 - Only base attributes are included in topology dataset, limiting usefulness of tree table
- The tree table collapses all nodes when a relationship is removed
 - When TBSM removes parent/child relationship, topology dataset is rebuilt
 - Refresh notification sent to tree table results in all nodes being collapsed, so it does not retain tree expansion between views

Known problems and limitations

A flat data set is one that doesn't contain relational information.

The topology or tree table widgets only access the base information.

With tree table widgets, if you configure it so that the tree is expanded, it will collapse again once refreshed. You can't specify column width and column order and expect it to persist.

Known problems and limitations

- Sorting/filtering delayed after updates to column used in sort/filter
 - Provider schedules refresh event – by default refresh then occurs within the next 30 seconds, depending on whether refresh is already queued
 - A property can be changed to modify the 30 second value, with 0 removing the delay
 - Column value updates are sent immediately and queued for the client
 - Resulting table can look unsorted and/or unfiltered until next refresh
 - With data fetchers running on a frequent interval, this could be a persistent issue
- Only status gauge directly supports TBSM status property
 - Other gauges require setting thresholds
 - For example, TBSM has a numeric rule with thresholds
 - Use analog gauge to display the rule value
 - Thresholds must be set in gauge Quick Edit to reflect thresholds set in TBSM
 - Allows setting different threshold values, but this would generally not be good, or might be by mistake

Known problems and limitations (continued)

Updates are pushed by TBSM. If you have a portal on screen and a DASH console on another screen, you might see about 30 seconds delay to when the update is made on the DASH console.

If you look at a table widget and sort on a particular column, a push to update might interfere, requiring the sort to be re-done.

The status gauge can support a TBSM status property.

No services being displayed

- Does the user have access to the expected services?
 - What are the TBSM user global and object level settings?
 - How is the remote UI data provider connection defined?If Single Sign on is being used, does DASH user have required credentials/roles on TBSM Dashboard server?
If no Single Sign on, are the credentials sufficient for the user that created the remote provider connection?
- Is the configured dataset a template-based dataset?
 - Check the TBSM console to ensure the template has services tagged with the template
 - If a primary template dataset, there must be services tagged with the template as the primary template
- Does the name specified for parameter TBSM Service exist?
 - Is the service instance name correctly typed as the parameter value?
 - Display name cannot be used, only service instance name.

No services being displayed

If you have single sign on configured and you set this up in the Connection document, for any dashboard that requests data through the TBSM data provider, the user name will be applied on the TBSM side to see if the authorizations are sufficient.

Troubleshooting: Unexpected status value

- Status displayed on DASH page does not match status shown in TBSM scorecard
 - Overall service status should generally stay in sync – issue could happen for a status associated with a specific TBSM rule
 - Status will not be synchronized due to latencies built into different products/layers
 - TBSM client model, for example, service tree and service viewer, refresh on a fixed interval
 - REST layer has some built in latency for optimization of performance
 - Do thresholds defined for DASH gauge match the thresholds defined by the TBSM rule?
 - This could cause mismatch between TBSM scorecard and DASH gauge.
- Status displayed in DASH table does not match status shown in DASH gauge
 - Allow for slight timing differences in widget applying updates
 - Do thresholds defined for DASH gauge match the thresholds defined by the TBSM rule?
 - The table widget is displaying the status value provided in the dataset by TBSM.
 - The gauge is calculating a status based on thresholds set in DASH Quick Edit.

Troubleshooting: Unexpected status value

This slide describes the synchronization issue and that you can override thresholds.

Best Practice: Use of generic provider ID for connections

- Do not use the default provider ID
 - Default DASH provider ID when defining a connection includes hostname
 - When pages move to another system, pages will have the wrong provider ID
 - Configuration of each widget on each page must be updated
- Use a “custom” provider ID for the connection, for example TBSM.DataProvider
- use this same ID on target system when pages are moved
- no update required to pages
- Limitations exist that will be addressed in DASH and TBSM Fixpacks
- More information
- Dashboard Best Practices Document
- https://www.ibm.com/developerworks/community/files/form/anonymous/api/library/b6f53616-38e7-47ca-bc35-4cc465e5fcb7/document/d8fadb7c-291e-4192-93ab-7d66ad096333/media/Dashboards-Best%20Practices_v1.2.pdf

Best Practice: Use of generic provider ID for connections

When you create the Connection document, it is best not to tie it to the host name but keep it generic. That makes it more portable. Then when you export pages and import into another environment, the pages will be usable immediately. Otherwise, you will have to reconfigure each of the pages.

You can set up provider IDs that match another environment, export, and then import into another environment without a problem.

Best Practice: Synchronizing DASH status with TBSM status

- Where possible, use the Status gauge in DASH
- For the gauge value select **TBSM Status**
 - Metric with thresholds in TBSM, shows as Metric Display Name (Status)
 - Gauge will be set based on status attribute
 - Configuration of the gauge will not prompt for threshold settings
 - For all other gauges, a metric value is displayed and thresholds must be set
- It is the responsibility of the page creator to set the same thresholds as defined in the TBSM rule

Best Practice: Synchronizing DASH status with TBSM status

This slide describes setting thresholds.

Reference Materials

- TBSM Wiki on developerWorks
 - <https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/Tivoli%20Business%20Service%20Manager1/page/Home>
 - Check out the Advanced Topics page
- TBSM Development / Internal Wiki
 - <http://w3.tap.ibm.com/w3ki03/display/TBSMDev/Home>
 - Search topics: **twl debugging**
- JazzSM on developerWorks
 - <https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=69ec672c-dd6b-443d-add8-bb9a9a490eba>
- Service Management Connect
 - <https://www.ibm.com/developerworks/servicemanagement/>
- TBSM Wiki on developerWorks – Advanced Topics
 - <https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Business%20Service%20Manager1/page/Advanced%20Topics>

Reference Materials

These are useful links.

<https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Business%20Service%20Manager1/page/Advanced%20Topics>

Exercise 1
Creating the dayTrader logo custom image widget

Exercise 2
Creating the dayTrader Revenue Detail page

Exercise 3
Creating the dayTrader Revenue Status page

Exercise 4
Linking pages with wires

Exercise 5
Testing the dashboard design

Complete Unit 8 Exercises 1 through 5 in the exercise guide.

Summary

Now that you have completed this unit, you can perform the following tasks:

- Describe the function and relationships of templates, rules, and services in Tivoli Business Service Manager
- Describe the Tivoli Business Service Manager primary architectural components
- Describe the Tivoli Business Service Manager data provider architecture
- Describe the general properties of the Tivoli Business Service Manager datasets
- Use Tivoli Business Service Manager data sets to show business service status and key performance indicators in business dashboards

Unit 9 Networks for Operations Insight dashboards

IBM Training



Unit 9 Networks for Operations Insight dashboards

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this unit, you review the integration points between Networks for Operation Insight and DASH. You review, at a high level, the Networks for Operation Insight architecture and the ways in which network event data can be used with dashboards.

Objectives

When you complete this course, you can perform the following tasks:

- Describe the major functions of Netcool Operations Insight
- Describe the major functions of Networks for Operations Insight
- Describe the capabilities of the Network Health Dashboard

Lesson 1 Networks for Operations Insights overview

IBM Training



Lesson 1 Networks for Operation Insights overview

Netcool Operations Insight base features

- Event search
 - Applies the search and analysis capabilities of Operations Analytics Log Analysis to events that are monitored and managed by Tivoli Netcool/OMNibus
 - ObjectServer events used by Log Analysis and indexed for searching
- Event Analytics
 - Netcool/Impact performs statistical analyses of Tivoli Netcool/OMNibus historical event data
- IBM Connections integration
 - Netcool/Impact enables social collaboration through IBM Connections by automatically providing updates to key stakeholders

Netcool Operations Insight base features

IBM Netcool Operations Insight uses real-time alarm and alert analytics, which are combined with broader historic data analytics. Netcool Operations Insight uses the fault management capabilities of IBM Tivoli Netcool/OMNibus and IBM's leading big data technologies within IBM Operations Analytics Log Analysis, providing powerful event search and historical analysis in a single solution.

The main features of the base solution are as follows:

- Event search

Combines the features of Netcool/OMNibus for comprehensive event management with the search capabilities of IBM Operations Analytics Log Analysis.

- Event Analytics

Netcool/Impact analyzes events from the Netcool/OMNibus event archive database. The analysis looks for events that repeat and events that are related.

- IBM Connections integration

Netcool/Impact provides the integration to IBM Connections. The integration is used to automatically post information on an IBM Connections forum.

Networks for Operations Insight

- Optional feature that can be added to a deployment of the base Netcool Operations Insight solution
 - Integrates IBM Tivoli Network Manager and IBM Tivoli Netcool Configuration Manager
- Topology search
 - An extension of the Networks for Operations Insight feature
 - Applies the search and analysis capabilities of Operations Analytics Log Analysis to give insight into network diagnostics
- Network Health Dashboard
 - Monitors a selected network view
 - Displays device and interface availability within the selected view
 - Reports on performance by presenting graphs, tables, and traces of KPI data for monitored devices and interfaces

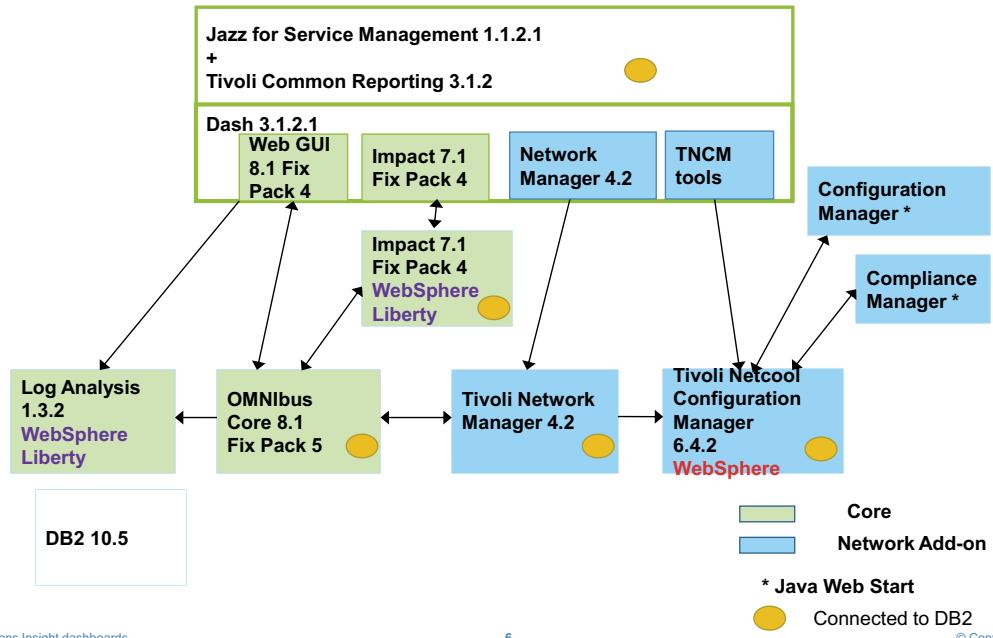
Networks for Operations Insight

Netcool Operations Insight consists of a base solution for managing and analyzing application monitoring environments and also an optional extension that is called Networks for Operations Insight, which widens the scope to include monitoring and analyzing network events.

Netcool Network Manager now uses DASH for its user interface.

You can use TDI or Impact to communicate with the network database.

Netcool Operations Insight 1.4 with Network option



Netcool Operations Insight 1.4 with Network option

This slide illustrates the component architecture of the solution. The components in blue, on the left, represent the base solution and the components in green, on the right, represent the network option.

Network Operations Insight requires Jazz for SM 1.1.2.1 and it includes TCR 3.1.2.

DASH server is required to support the web GUI.

Impact is a Blaze enabled console, so you have to set up Single Sign-on for the Impact console.

Add-on products Network Manager and TNCM also require a DASH server.

Impact Server is a separate component that can be installed on the same server.

Log Analysis is a separate component with its own Liberty based interface.

Performance Insight and OMNIBus are integrated with Netcool Operations Insight.

Additional optional components

Network Performance Insight

- Flow-based network traffic performance monitoring system
- Provides comprehensive and scalable visibility on network traffic with visualization and reporting of network performance data for complex, multivendor, multi-technology networks

IBM Alert Notification

- Provides instant notification of alerts for any critical IT issues across multiple monitoring tools
- Gives IT staff instant notification of alerts for any issues in your IT operations environment

IBM Runbook Automation

- Investigate and delegate problems faster and more efficiently
- Diagnose and fix problems faster and build operational knowledge
- Easily create, publish, and manage runbooks and automations
- Keep score to track achievements and find opportunities for improvement

Additional optional components

You can also set up IBM Network Performance Insight as part of your Netcool Operations Insight solution to monitor network traffic performance, and you can integrate with further solutions such as IBM Alert Notification and IBM Runbook Automation.

With Alert Notification you configure your OMNIbus server to forward certain events to these two products. Can use in voice mail, email, or text.

Base solution components

- Tivoli Netcool/OMNIbus core V8.1.0.5
- Gateway for JDBC
Used to populate event archive database
- Tivoli Netcool/OMNIbus Web GUI V8.1.0.4
With Netcool Operations Insight extensions
- Netcool/Impact V7.1.0.4
With Netcool Operations Insight extensions
- IBM Operations Analytics Log Analysis V1.3.2 Standard Edition
- Tivoli Netcool/OMNIbus Insight Pack V1.3.0.2 for IBM Operations Analytics Log Analysis
- Gateway for Message Bus V7.0
- Jazz for Service Management V1.1.2.1

Base solution components

The Netcool Operations Insight solution consists of the products that are shown on this slide. Each of these products is available for purchase individually. The Netcool Operations Insight solution includes features that are not available with the individual products when you purchase the products individually. The extensions are not available when a product is purchased individually.

Optional network management components

- IBM Tivoli Network Manager V4.2
- Probe for SNMP
- Syslog Probe
- Network Manager Insight Pack V1.3.0.0 for IBM Operations Analytics Log Analysis
- IBM Tivoli Netcool Configuration Manager V6.4.2
- Network Health Dashboard V4.2

Optional network management components

This slide shows the products that comprise the network management option for Netcool Operations Insight.

Networks for Operations Insight installation considerations

- The base *Netcool Operations Insight* solution includes Tivoli Netcool/OMNibus V8.1, Netcool/Impact V7.1, and SmartCloud Analytics - Log Analysis V1.2, and Jazz for Service Management 1.1.2
- The *Networks for Operations Insight* feature includes Network Manager V4.1.1, Tivoli Integrated Portal 2.2, Tivoli Netcool/OMNibus V7.4, Web GUI, and Netcool Configuration Manager V6.4.1.
 - Extra configuration is required to set up the integration between the base solution and the Networks for Operations Insight feature.
- Tivoli Netcool/OMNibus Web GUI V7.4 and Tivoli Netcool/OMNibus Web GUI V8.1 must be installed in parallel to support the base solution and network feature.
 - Web GUI V7.4 requires the Tivoli Integrated Portal and Web GUI V8.1 requires DASH
 - DASH is required to take advantage of extra functions, including the Seasonality feature, that the Netcool Operations Insight V1.2 solution provides.

Lesson 2 Network health dashboards review

IBM Training



Lesson 2 Network health dashboards review

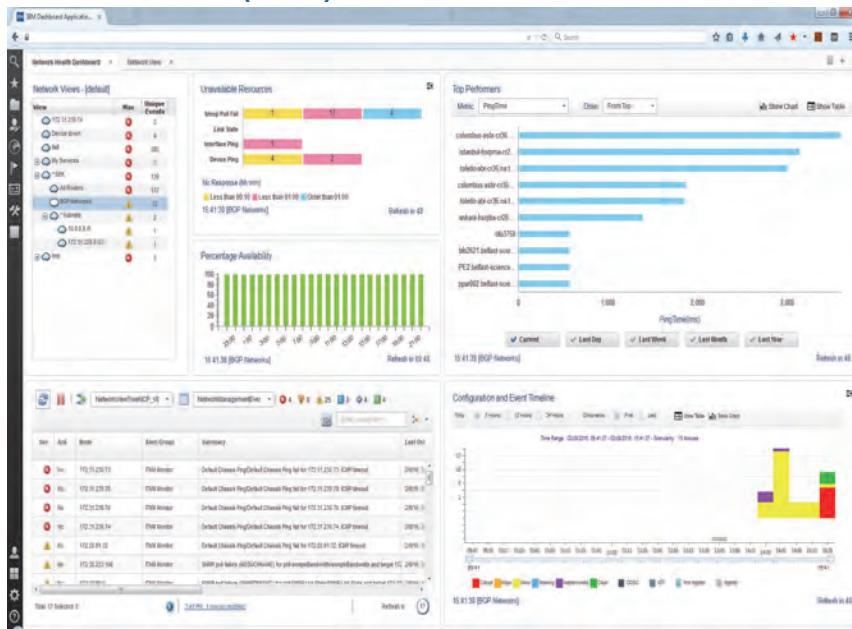
Overview

- Combines near real-time network health information from three products
 - Network Manager
 - Network Configuration Manager
 - Netcool/OMNIbus
- The dashboard is broken into reusable widgets describing the state or health of the network
 - Network Availability
 - Network Polling Metrics
 - Network Event summary and details
 - Configuration Correlation

Overview

The Network Health Dashboard monitors a selected network view, and displays device and interface availability within that network view. It also reports on performance by presenting graphs, tables, and traces of KPI data for monitored devices and interfaces. A dashboard time-line report on device configuration changes and event counts, enables you to correlate events with configuration changes.

Network Health Dashboard (NHD)



Networks for Operations Insight dashboards

13

© Copyright IBM Corporation 2016

Network Health Dashboard (NHD)

Monitor the Network Health Dashboard by selecting a network view within your area of responsibility, such as a geographical area, or a specific network service such as BGP or VPN. Then, review the data that appears in the other widgets on the dashboard. If you have a default network view bookmark that contains the network views within your area of responsibility. Then, the network views in that bookmark appear in the network view tree within the dashboard.



Hint: Recommended screen resolution of 1536 x 864 pixels or greater.

Network views

- Network topology views
 - All widgets are scoped on an area of responsibility as defined by a Network Manager View
- Purpose of the default bookmark
 - Gives user a choice for what views to include
 - More efficient than large network view tree
- Clicking a view
 - Publishes a NodeClickedOn event that other Dash widgets can consume
 - Includes the NM view ID and bookmark ID
 - Drives context for charts

The screenshot shows the Network Health Dashboard interface. On the left, there is a vertical toolbar with icons for search, star, folder, user, network, play, list, and tools. To the right of the toolbar is a tree view titled "Network Views - [default]". The tree structure includes nodes like "172.31.230.74", "Device down", "itall", "My Services", "SBK", "All Routers", "BGP Networks", "Subnets", "10.0.0.0 /8", "172.31.228.0 /22", and "test". The "BGP Networks" node is currently selected, indicated by a blue selection bar at the bottom of its row. A table to the right of the tree provides statistics for each view, showing the maximum number of events and unique events. The table has columns for "View", "Max", and "Unique Events".

View	Max	Unique Events
172.31.230.74	2	
Device down	4	
itall	300	
My Services	3	
SBK	139	
All Routers	137	
BGP Networks	32	
Subnets	2	
10.0.0.0 /8	1	
172.31.228.0 /22	1	
test	3	

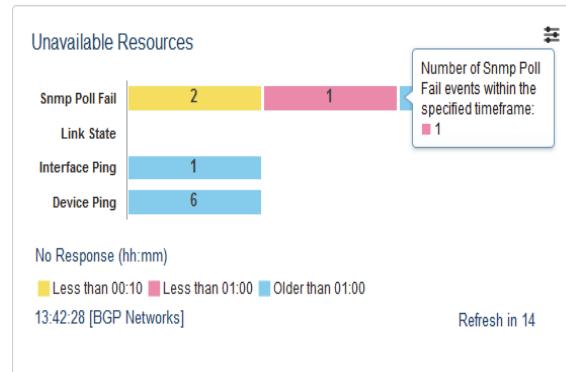
Network Views

In the Network Health Dashboard, select a network view from the network view tree in the Network Views at the upper left. The other widgets update to show information based on the network view that you selected. In particular, the Top Performers widget updates to show overall availability of chassis devices in network view.

Network Availability (1)

Unavailable Resources

- Temperature gauge of the responsiveness of the network
 - Device and Interface ping events
 - SNMP Poll Failures and SNMP Link State
- Note:** Events severity **clear** are not in count tally
- Correlated data from
 - Network Manager polling framework
 - Netcool/OMNIbus
- Preferences
 - User can choose relevant polls
- Drill down
 - Selecting a segment updates any Event viewer wired for *NodeClickedOn* events



Network Availability (1)

To determine the number of unavailable devices and interface alerts, use the following sections of the chart and note the colors of the stacked bar segments and the number inside each segment.



Note: By default, all of the bars described here are configured to display. However, you can configure the Unavailable Resources widget to display only specific bars. For example, if you configure the widget to display only the Device Ping and the Interface Ping bars, then only those bars are displayed in the widget.

SNMP Poll Fail

Uses color-coded stacked bars to display the number of SNMP Poll Fail alerts within the specified time frame.

SNMP Link State

Uses color-coded stacked bars to display the number of SNMP Link State alerts within the specified time frame.

Interface Ping

Uses color-coded stacked bars to display the number of Interface Ping alerts within the specified time frame.

Device Ping

Uses color-coded stacked bars to display the number of Device Ping alerts within the specified time frame.

Color coding of the stacked bars is as follows: Click any one of these bars to show the corresponding alerts for the devices and interfaces in the Event Viewer at the bottom of the Network Health Dashboard.

Network Availability (2)

- Percentage Availability
 - Trend for the previous 24 hours
 - Shows how responsive Devices are to Device Chassis Ping poll definition
- Correlated data from Network Manager Aggregated Polling Data



Network Availability (2)

You can monitor overall availability of chassis devices within a selected network view by using the Percentage Availability widget.

The Percentage Availability widget displays 24 individual hour bars. Each bar displays a value, which is an exponentially weighted moving average of ping results in the past hour; the bar appears only on the completion of the hour.

The bar value represents a percentage availability rate rather than a total count within that hour. The color of the bar varies as follows:

- Green: 80% or more.
- Orange: Between 50% and 80%.
- Red: Less than 50%.

Polling Metrics

Network Manager Polling Chart

- Known as the Top Performers widget
- Display the 10 maximum or minimum values for a selected Poll metric
- Show current or aggregated historical poll data
- Data can be shown in a bar graph or table
- Correlated data from
 - Network Manager Aggregated Polling Data
- Drill down
 - Clicking the TopN bar returns all stored trace data for network entity



Networks for Operations Insight dashboards

17

© Copyright IBM Corporation 2016

Polling Metrics

Polling Metrics

Widget gives a view of the Top N performing devices based on a Network Manager poller policy over a defined time range.

Click one of the following icons to specify which current or historical poll data to display in the main part of the window. This icon updates the data regardless of which mode is being presented: bar chart, table, or time trace.

Current

Click this icon to display current raw poll data. When in time trace mode, depending on the frequency of polling of the associated poll policy, the time trace shows anything up to 2 hours of data.

Last Day

Click this icon to show data based on a regularly calculated daily average.

- In bar chart or table mode, the top 10 highest or lowest values are shown based on a daily exponentially weighted moving average (EWMA).
- In time trace mode, a time trace of the last 24 hours is shown, based on the average values.

In the Last Day section of the widget, EWMA values are calculated by default every 15 minutes and are based on the previous 15 minutes of raw poll data. The data that is presented in this section of the widget is then updated with the current EWMA value every 15 minutes.

Last Week

Click this icon to show data based on a regularly calculated weekly average.

- In bar chart or table mode, the top 10 highest or lowest values are shown based on a weekly exponentially weighted moving average (EWMA).
- In time trace mode, a time trace of the last 7 days is shown, based on the average values.

In the Last Week section of the widget, EWMA values are calculated by default every 30 minutes and are based on the previous 30 minutes of raw poll data. The data that is presented in this section of the widget is then updated with the current EWMA value every 30 minutes.

Last Month

Click this icon to show data based on a regularly calculated monthly average.

- In bar chart or table mode, the top 10 highest or lowest values are shown based on a monthly exponentially weighted moving average (EWMA).
- In time trace mode, a time trace of the last 30 days is shown, based on the average values.

In the Last Month section of the widget, EWMA values are calculated by default every 2 hours and are based on the previous 2 hours of raw poll data. The data that is presented in this section of the widget is then updated with the current EWMA value every 2 hours.

Last Year

Click this icon to show data based on a regularly calculated yearly average.

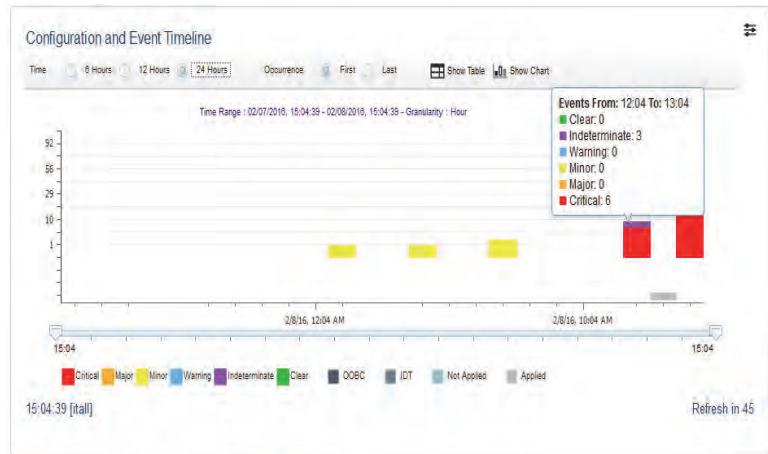
- In bar chart or table mode, the top 10 highest or lowest values are shown based on a yearly exponentially weighted moving average (EWMA).
- In time trace mode, a time trace of the last 365 days is shown, based on the average values.

In the Last Year section of the widget, EWMA values are calculated by default every day and are based on the previous 24 hours of raw poll data. The data that is presented in this section of the widget is then updated with the current EWMA value every day.

Configuration and Event Timeline (1)

Event timeline

- Shows event distribution over time
- Events for device and interfaces in a network view
- Time options
 - First or last occurrence of event
 - Granularity
 - [Default] 15 mins over 6 hours
 - 30 mins over 12 hours
 - 60 mins over 24 hours
- Correlated data from
 - Netcool/OMNIbus Events
- Preferences
 - User can hide event severities from user preferences
- Event only chart
 - When NCM is not integrated with Network Manager



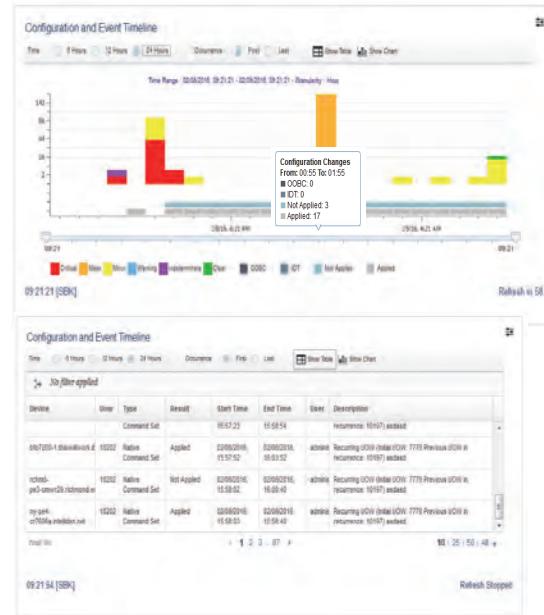
Configuration and Event Timeline (1)

You can display a timeline showing, for all devices in a selected network view, device configuration changes and network alert data over a time period of up to 24 hours. Correlation between device configuration changes and network alerts on this timeline can help you identify where a network issue was likely caused by configuration changes.

Configuration and Event Timeline (2)

NCM Configuration rug chart

- Configuration events in the scope of a Network View
- Shows atomic configuration per device
- Chart gives binary indication of a configuration change
- Correlated data from
 - Integrated Network Configuration Manager
 - Chart automatically shows configuration events
- Configuration types shown
 - Out Of Band Changes (OOBC)
 - ITNCM Device Terminal (IDT)
 - Applied or not applied changes
- Drill Down
 - Configuration events details per device
 - Dynamic filter
 - Correlation to Unit of Work in Network Configuration Manager



Networks for Operations Insight dashboards

19

© Copyright IBM Corporation 2016

Configuration and Event Timeline (2)

Configuration changes displayed in the Configuration and Event Timeline can be any of the following items. Move your mouse over the configuration change bars to view a tooltip that lists the different types of configuration change made at any time on the timeline.

Changes that are managed by Netcool Configuration Manager

These changes are made under full Netcool Configuration Manager control. The timeline differentiates between scheduled or policy-based changes, which can be successful (Applied) or unsuccessful (Not Applied), and one-time changes made by using the IDT Audited terminal facility within Netcool Configuration Manager.

Applied

A successful scheduled or policy-based set of device configuration changes made under the control of Netcool Configuration Manager.

Not Applied

An unsuccessful scheduled or policy-based set of device configuration changes made under the control of Netcool Configuration Manager.

IDT

Device configuration changes made by using the audited terminal facility within Netcool Configuration Manager that allows one-time command-line based configuration changes to devices.

Unmanaged changes

OOBC

Out-of-band-change. Manual configuration change that is made to device where that change is outside of the control of Netcool Configuration Manager.

Events are displayed in the timeline as stacked bars, where the color of each element in the stacked bar indicates the severity of the corresponding events. Move your mouse over the stacked bars to view a tooltip that lists the number of events at each severity level. The X-axis granularity for both events and configuration changes varies depending on the time range that you select for the timeline.

Instructor demonstration

- Networks for Operation Insight



Unit 10 Exporting and importing customizations

IBM Training



Unit 10 Exporting and importing customizations

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this unit, you review the dashboard design process and summarize the responsibilities and tips for delivering a successful customer proof-of-concept.

Lesson 1 Exporting and importing dashboard customizations

IBM Training

IBM

Lesson 1 Exporting and importing dashboard customizations

Objectives

- In this unit, you learn how to export and import DASH components that you create

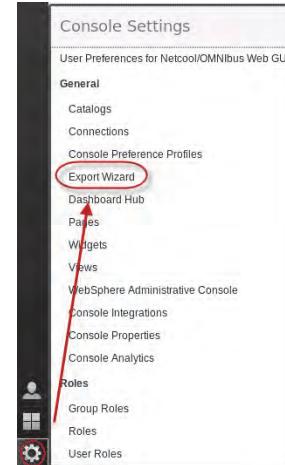
Export and import scope

The following dashboard elements can be exported and imported:

- Custom dashboards and customized system page elements including:
 - Dashboard name and layout
 - Widget configurations
 - Views
 - Events and wires
 - Access permissions
 - Navigation structure
- Custom authorization roles, including:
 - Role name, creation date, and update date
 - Role mapping information in relation to users and groups
 - Associated role preference for relevant console preference profile
- Console properties and customization properties, including:
 - Transformations
 - Themes and images

Exporting dashboard customizations

- Customizations can be exported for migration to another DASH environment or as a backup
- Export using the GUI-based Export Wizard task or the **consolecli.sh** command line script
 - There is no GUI-based import wizard tool
 - The Export Wizard tool shows the page name correlated with the unique page ID
- The **consolecli.sh** script is located in the **\$JAZZHome/ui/bin** directory
 - Example: **/opt/IBM/JazzSM/ui/bin/consolecli.sh**



Export wizard example

- Select the **Console Settings > General > Export Wizard** task
- Step through the wizard and select **Choose which console settings and customizations to export** and to download the package to the console workstation
- Select to export (if available) pages, console preference profiles, and views

Export Options

What would you like to export?

All Dash settings and customizations

Choose which console settings and customizations to export.

Export package name:

Download package to my desktop on completion.

Detail Options

Select the customizations settings to include in the export package.

Select the items to export:

- Console Preference Profiles
- Custom and SystemCustomized Views
- Custom and SystemCustomized Pages

Export wizard example (continued)

- Depending on the export selections, you are prompted to choose which pages, views, and conference preference profiles are to be included in the export archive
 - Views configured in selected profiles are automatically included in the archive.
 - The output archive filename defaults to the form **DASH_export_<yyyymmdd>.zip**

The screenshot shows the 'Export wizard example (continued)' interface. It consists of three main panels:

- Profiles to Export:** A table with columns 'Select' and 'Profile Name'. One row is selected, labeled 'mobileDashboardUsersProfile'.
- Views to Export:** A table with columns 'Select' and 'View Name'. One row is selected, labeled 'mobileDashboardUsersView'.
- Pages to Export:** A table with columns 'Select' and 'Name'. Four rows are listed:
 - Impact Policy Variables Page (selected)
 - Sales By Location (selected)
 - Sales Manager Contacts (selected)
 - Test Custom Content (not selected)

Export command line examples

- To list pages that are available for export:
`consolecli.sh ListPages --customizePages true`
- Output is a list of pages, in the form:
`com.ibm.isclite.global.custom.module-SPSVS-com.ibm.isclite.admin.Freeform-navigationElement.pagelayoutA.modified.<pageID>`
- The <pageID> value is unique for each page in the DASH server
- To export a specific page:
`consolecli.sh ExportPage --uniqueName <pageID1>,<pageID2> --username <userID> --password <password>`
- To export all customizations:
`consolecli.sh Export --username <userID> --password <password>`

Import command line examples

- The import archive must be in the \$JAZZHOME/ui/input directory, with the file named **data.zip**
 - Example: `cp DASH_export_04012016.zip /opt/IBM/JazzSM/ui/input/data.zip`
- To import the data archive, use the following command form:
`$JAZZHOME/ui/bin/consolecli.sh Import --username <userID> --password <password>`
 - Example: `/opt/IBM/JazzSM/ui/bin/consolecli.sh Import --username smadmin --password object00`
- Imported data is merged with the target DASH environment
- To roll back an import, use the following command form:
`$JAZZHOME/ui/bin/consolecli.sh Import --rollback ALL --username <userID> --password <password>`

Exercise 1 Exporting DASH customizations

Exercises

Complete Unit 10 Exercise 1 from the exercise guide.

Student demonstrations

- Exporting DASH customizations



Summary

- When you complete this unit, you will know how to export and import DASH components that you create



IBM Training



© Copyright IBM Corporation 2016 All Rights Reserved.