What's New in Tivoli Directory Integrator Version 6.1.1

Authors: Eddie Hartman
Johan Varno

Contents

Introduction

As with previous versions, the following changes and enhancements are targeted at TDI solution developers and bundlers. Of course, improvements to the usability of the Administration and Monitoring Console (AMC) and Action Manager (AM) have also been done with TDI administrators and bundlers in mind.

This release is a "Modification Release" which means it is a new version of the existing release (6.1) – hence the number 6.1.1. As a result, it contains items that do not impact backwards compatibility; With the exception of the Config version number, which is now in step with that of the system itself: version = 6.1.1. As was the case with 6.1 and 6.0 (both introduced changes to the format of a Config), previous releases will not be able to open files created with a newer version.

One new feature that will please our bundlers[1] and users alike is that TDI now runs on i5/OS and HP Itanium. TDI also boasts a number of new Autonomic Computing features and components to let you plug your solutions into an AC architecture. This release provides an example of how to monitor TDI from IBM Tivoli Monitoring/Omegamon. In addition, the new Secure Remote CommandLine Connector helps to securely extend the reach of monitoring even further.

For developers we've added the ability to Debug/Step your AssemblyLines on remote servers, so you can sit in the comfort and safety of your laptop and interactively test your ALs in-place on another server, on another platform. Maintenance and testing of distributed servers is made even easier with the usability enhancements to the TDI Administration and Monitoring Console (AMC) and Action Manager (AM). This includes the ability for TDI developers to define the *published interface* to your solution directly from the Config Editor (CE) via the new Configs > Solution Instance feature. Plus you will find an example of how to set up the JMS Connector to work with other JMS-compliant message queue technologies.

If you are using the Delta Engine to detect changes in files or other sources without change-notifications, then you will be happy to see a dramatic performance improvement when processing large data sets with few modifications.

Of course there are a number new components like the SAP Change Detection Connector, TAM Connector and TIM DSML v2 Connector, plus many more exciting enhancements to explore and leverage.

[1] These include at the time of writing TIM, TAM, CCMDB/TADDM, WebSphere RFID Information Center and Domino/Lotus Connections.

1 New features

1.1 Platform support for i5/OS and HP Itanium

TDI now runs natively on i5 (previously known as OS/400) V5R4. This includes all TDI components except the Config Editor itself. The installer will not bundle the 1.5 JRE or the embedded WAS Express v6.0.2, so detection/setup of these must be done manually.

Similar support is also provided for the HP Itanium platform.

1.2 Debugger/Stepper Enhancements

1.2.1 New debugging Run modes

The "bug" button to start the AssemblyLine Stepper (Debugger) has been replaced by a Run Mode drop-down that offers all previously available options, plus a new one that lets you *run the AL and revert to debug in case of error*. This selection is called "Step (Break on error)" and is also the new default behavior of the "run" button now.

Note: Since the Debugger/Stepper only allows an AL Pool size of 1 (one), this means that if you wish to launch a Server Mode Connector-based AssemblyLine with a greater pool size, you must select and use the Standard Run mode.

1.2.2 Remote Debugging

The AL Stepper/Debugger now supports interactively running and testing a Config that is running on a remote TDI server – which can be running on the same machine, or across the infrastructure on a different platform. This functionality is available whenever a Remote Config is loaded into the Config Editor.

1.3 ITM/Omegamon Integration with JMX

An example is provided in the examples\Tivoli_Monitoring folder of the TDI installation directory that illustrates how TDI can be integrated with ITM using the JMX management interface. TDI servers and TDI configs can both be monitored by ITM, and operational or custom events are sent from TDI into the ITM infrastructure.

You will find the jar-file of the Universal Agent, as well as a readme.txt that contains instructions on how to set up and exploit this component.

1.4 Enable sharing of Iterator connection

Re-using connections allows you to limit the resources used by an AssemblyLine, and is done by clicking on the **Inherit From** button at the lower right-hand corner of a Connector's **Config > Connections** tab. At the top of the resulting drop-down selection dialog are the other Connectors in this AL. By choosing one of these, you are instructing these components to share the same connection to the target system.

This feature is now supported for Connectors in Iterator mode as well. Although this connection re-use feature has been available in previous TDI versions, the behavior has not been predictable when re-using Iterators.

Note that this sharing of connections between Connectors in a single AssemblyLine is not to be confused with the Global Connector Pool feature which is available via the Connector **Pool** tab.

1.5 Autonomic Computing (AC)

Although systems management via SNMP and IBM Tivoli Monitoring communications is a common need, many IT resources still resist insight. IBM's vision of Autonomic Computing (AC) rests on a foundation of state-awareness, particularly the state of mission critical systems and services, as well

as their dependencies on each other; on sub-systems and partitions; even on metal and silicon[2]. Many applications, data sources and networking products support this kind of direct monitoring (as TDI does via JMX, SNMP and other protocols/technologies). For those that don't, TDI provides functionality to let you reach into the darker corners of the infrastructure.

Often operational data can be retrieved via protocols or direct API calls, collected, interpreted and then delivered upstream to monitors and reporting systems[3]. Other times information about security, performance and availability must be gleaned from log files, requiring a battery of parsers to deal with each proprietary log format. Furthermore, the shear volume of events occurring in a modern, 24-7 infrastructure mandates an agile and robust technology to capture, translate and securely transfer this information.

Part of IBM's Autonomic Computing strategy is to harmonize this event traffic by defining a standard Common Base Event (CBE) format, as well as services for dependably correlating and switching these event streams in real-time. However, you need tooling and integration services to help you reach this vision. Fortunately TDI lets you quickly and securely leverage AC standards and technologies in your own infrastructure.

Previous versions of TDI already provided support for the CBE format through the CBEGenerator FC and XML parsing features. TDI version 6.1.1 includes a number of new components and technologies to take you even closer to the goals and benefits of AC by simplifying access to these services, as well as extending the "operational vocabulary" of your existing IT systems.

For a closer look at IBM Autonomic Computing technologies, check out this content on DeveloperWorks: http://www-128.ibm.com/developerworks/autonomic/solins.html.

1.5.1 Autonomic Generic Log Adapter (GLA) Connector

The Autonomic Generic Log Adapter is a flexible technology that is primarily used to read/parse log files and send this information on to other systems for further processing (such as LTA - Log and Trace Analyzer). GLA is a standalone application that supports more than 140 log formats and sends data (in CBE format) off to other systems using the Autonomic Remote Agent Controller (RAC – see the next section).

The GLA engine is included with TDI so that you can receive CBE-formatted messages using its rich set of log adapters. This information is easily mapped into your AssemblyLines for handing and propagation using the GLA Connector. There is an example of how to configure and use this component in the examples/GLAConnector sub-folder of your TDI installation. Note that you will either need to copy this directory under your TDI solution directory, or edit the adapter configuration file (called "adapter.adapter") to specify which directory to use[4]. Once the setup is complete, simply open the example Config (GLAConnectorExample.xml) and then connect and read using the Connector.

GLA also has its own Eclipse tooling if new parsers need to be written, although this is not part of the bundle with TDI.

1.5.2 Autonomic Remote Agent Controller (RAC) Connector

Remote Agent Controller (RAC) is a lightweight "message switch/queue" that manages communications with tools like GLA (described in the previous section) and other Autonomic Computing (AC) *agents.* RAC receives status data (aka "events") from remote agents, while allowing applications like Log & Trace Analyzer (LTA) to subscribe to those that fit specific profiles. RAC then ensures that subscribers receive the desired event stream. However, this is synchronous communication and RAC does not persist event data. Enter TDI.

---

[2] This is where IBM Change and Configuration Management Database (CCMDB) comes in handy, with its Tivoli Application Dependency Discovery Manager (TADDM) that lets you detect and migrate the details of how your infrastructure hangs together. This information is leveraged by the IBM ITSM suite of products to help you plan for change with fewer surprises and unexpected knock-on effects.

[3] As TDI does in its bundle with Tivoli Directory Server.

[4] The XML text value you need to change is this one:
<pu:Property propertyName="directory" propertyValue="**C:\Program Files\IBM\TDI\V6.1.1\examples\GLAConnector**"/>

TDI provides a RAC Connector allowing you to easily manage connections to RAC, send and receive Common Base Event (CBE) messages and transform/transfer events as desired. This new functionality allows TDI solutions to play directly in an AC environment acting as remote agents, subscribing/dispatching specific events and even persisting status information for later evaluation and consumption.

Two example scenarios that the TDI RAC Connector is designed to handle are:

- Sending data (in CBE format) to RAC, thereby augmenting the capabilities of adapter/sensor technology like GLA, and using TDI to create downstream data to LTA or other event consumers;
- Receiving CBEs from GLA where GLA is used in a stand-alone environment together with RAC (i.e. not embedded in TDI).

1.5.3 Autonomic Active Correlation Technology (ACT) engine

Active Correlation Technology (ACT) is a technology designed to build and execute rules for real-time correlation of events, particularly those events conforming to the Common Base Events (CBE) specification[5].

TDI provides the ACT engine embedded in the TDI server and allows you to feed it with Common Base Events using the ACTConnector. ACT can in turn be configured to pass desired events on to subscribers – including other ACTConnectors set up in Iterator mode.

You will find an example of how to use TDI with ACT in the examples\ACT sub-folder of your TDI installation directory. There is also more literature on ACT here:

http://www-128.ibm.com/developerworks/autonomic/library/ac-acact/index.html

1.6 Integration with Service Component Architecture (SCA)

An example is provided under the examples\sca folder of your TDI installation directory that illustrates how to publish a TDI AssemblyLine into an SCA infrastructure and the WebSphere Integration Developer (WID) tooling. A small Java application uses the TDI server API to communicate with the AssemblyLine, and an SCA descriptor is provided that defines the bindings into the AL.

1.7 TIM DSML v2 Connector

This Connector, which was removed from the TDI 6.1 release, has been re-introduced with the name "TIM DSML v2 Connector" to reflect its application to IBM Tivoli Identity Manager. Included also are the libraries (e.g. Castor) and TIM JNDI Driver required to provide this functionality.

Note that this change in name is to highlight the difference between this component, which speaks the DSML dialect used by TIM, and the "true" DSML v2 Connector already shipped with TDI[6].

1.8 SAP Change Detection Connector

There is a new SAP ALE IDoc Connector that enables TDI to hook into the outbound queue of SAP and "subscribe" to changes. This allows you to Iterate over changed data coming from SAP, although you will need to use TDI XML parsing components in order to interpret the outbound IDocs returned.

Furthermore, there are examples in the documentation on how to reach both SAP HR and CUA (Central User Administration) data; however, any information in SAP can be reached.

---

[5] At the time of this writing, the Eclipse-based ACT configuration tooling is only available for use by customers and partners working on projects together with the IBM Autonomics Computing team.

[6] Prior to TDI 6.1, this Connector was named the "DSML v2 Connector". Note that no re-configuration is needed if you have used this Connector under its old name. Automatic migration is performed by the system when it detects an older Config version.

1.9 New TAM Connector

This Connector enables TDI to directly read and write TAM account information, and is based on the same component used for the TIM adapter. The TAM Connector provides access to TAM Users, Groups, Policies, Domains, SSO Credentials, SSO Resources and SSO Resource groups.

Only TAM 6 is supported due to TAM 5.1 JRE support restrictions.

1.10 Castor "Java to XML" and "XML to Java" FCs

These powerful XML manipulation components were removed in TDI 6.1 due to Open Source liability reasons. The good news is that they are now back into TDI after a complete review, providing more flexible handling of complex XML. However, the libraries now included are not completely compatible with those found in pre-6.1 releases, so some adjustment may be necessary for migrated solutions using these.

2 Enhancements

2.1 Published AssemblyLine Initialization Parameters

In order give you finer configuration control when you call one AssemblyLine from another, the AL Operations tab has been enhanced to allow you to define initialization parameters. Any parameters defined for this AL are displayed in an AL FC or AL Connector that references this AssemblyLine[7]. Note also that you can set the *type* of each parameter to either String, Boolean, Number, Password or TextArea, which controls the type of input field displayed in the AL Connector and FC.

Whenever the called AssemblyLine is initialized (which happens when the calling AL Connector or FC initializes), these initialization parameters are accessible in your AssemblyLine via the *OpEntry (Operations Entry)* which carries one Attribute for each parameter. As a result, if you want to set component parameters with one of these initialization values you can simply use Expressions.

For example, if you have an initialization parameter called "UseURL", then you could tie this to the LDAP URL Parameter of an LDAP Connector by clicking on the label of that parameter (LDAP URL) and entering the following into the Expression field of the Parameter Dialog:

```
{op-entry.UseURL}
```

The OpEntry is also directly available to your scripts (e.g. the Prolog – Before Initialization Hook) by retrieving it from the AssemblyLine (task):

```
var opEntry = task.getOpEntry(); // get OpEntry from AL
thisConnector.connector.setParam("ldapUrl","Use.URL"); // set param
```

The OpEntry is also available whenever the AssemblyLine is actually executed as well, and the information in it is maintained as long as the AL is running. At this time, the actually Operation used for the AL call is also kept in the OpEntry as an Attribute called "$operation".

2.2 Delta Engine performance improved

A new algorithm is implemented that increases the performance for data sources that typically yield few changes by dramatically increasing the speed of detecting unchanged data. Note that the Delta Engine is fully backwards compatible with TDI v.6.1, and this new behavior is switchable via the new "Faster algorithm" checkbox in the Delta tab.

2.3 Default case for Switch

The switch component now supports a default case that is executed if no other case is matched.

Note that unlike Switch-Case in popular development languages (like Java, C and C++), TDI Case components do not "fall through"; instead, they behave like a series of if-else statements. Only a single Case is activated during an AL cycle, so you do not need to use calls like system.exitBranch() to break out from the Switch.

2.4 Improvements to Hook Inheritance

The system has been upgraded to allow you to enable or disable an inherited Hook in the Config Editor without breaking inheritance of the Hook to the parent component.

2.5 Domino Change Detection Connector Simplified

The Domino Change detection has been harmonized with the other Change Detection Connectors (CDCs) with regards to configuration and operation. For example, the ""Delivery Mode" parameter has been replaced with one common to all the CDCs called "State Key Persistence". The settings "After

[7] This behavior was available in 6.1 as well, although you had to manually add an Operation witht he special name "$initialize".

read" and "End of cycle" will behave similarly to the old "Assured once and only once delivery" value, with "End of cycle" ensuring that that Iterator State Key is only persisted to the System Store at the end of a successful AL cycle.

Note that the "Manual" setting for "State Key Persistence" will perform better (still behaving like the old "Normal assured delivery" did) but will now also allow some repetition of documents; Just as the previous parameter setting of "Normal assured delivery" did if the AssemblyLine aborted before the Iterator State was manually persisted. This is done with the following script call (given your AL Connector is called "DominoChangeALconn"):

```
DominoChangeALconn.connector.saveStateKey();
```

It is important to note that the order in which the Domino Change Detection Connector has always delivered changed documents is not in *change-timestamp* sequence because the Domino API doesn't provide that functionality. However, this updated component now offers a "Deliver Sorted" flag that will cause the Connector to perform local sorting of returned documents before continuing with the Iteration.

Note also that Domino allows non-existent users to be added to groups: something that is not supported in many other directories. As a result, it is best practices to either handle change propagation of users first, followed by groups – in other words, as two separate processes – or if you handle them both in one go, to re-apply any group changes that failed once all users are in place.

2.6 Documentation of JMS interface to Sonic MQ

An example is provided that illustrates how to configure the JMS Connector to work with different JMS systems; in this case, with Sonic MQ. While TDI has three different JMS-based Connectors – IBM WebSphere MQ, IBM WebSphere MQ Everyplace (MQe) and the JMS Connector – this example shows you how to leverage the JMS Connector parameter called "JMS Driver Script" to implement the logic necessary to work with Sonic MQ.

Note that you can change this example to support any JMS-compliant message bus by simply updating the "JMS Driver Script" as needed.

2.7 DDL Statements for SystemStore Tables Externalized

The SQL Data Definition Language (DDL) statements needed to set up the required TDI SystemStore tables is now externalized to the Global/System properties, allowing you to customize these to fit any JDBC-compliant RDBMS.

2.8 Sharing Logs with Called ALs

Both the AssemblyLine Connector and the AssemblyLine FC (used to invoke other ALs either locally or on a remote Server) now offer a configuration checkbox called "Share Logging" that causes the *called* AssemblyLine to share the Log Appenders specified for the calling AL.

2.9 Enhancements to Config Monitoring

2.9.1 Solution Instance

There is a new feature called Solution Instance found in the Config Browser under the Config option (where settings for Auto-Start and solution-level Logging are configured). Here is where you can define both a short, mnemonic name for your Config, as well as the list of ALs and properties that should be visible in AMC.

Previously a Config was only referable as long strings that included the full xml path and filename. Now with support for shorter and more descriptive Instance names, the usability of AMC, Action Manager (AM) and the TDI CommandLine Interface utility (CLI) are improved. The ability for developers to define how their Configs should be "published" to AMC, makes configuration of this powerful console simpler and faster; as well as making AMC configuration independent of Config **f**ile naming and path/location.

2.9.2 AMC and AM Improvements

This latest revision of AMC is focused on improving user experience, starting with the new welcome screen that provides quick links to common functions. In addition, there is a Quick-discover feature for automatic creation of Config Views either based on the Solution Instance settings created by the Config developer, or just for quickly exposing all ALs and properties in order to get AMC up and running quickly. Furthermore, Action Manager (AM) panels for configuring Actions have been updated and simplified to be easier to use.

Another improvement is the support in AMC for AssemblyLine Operations, allowing you to start ALs with a specific Operation selected, and then set up of any Input Attributes defined for that Operation. Furthermore, several additions to the monitoring screen (including quick commands and better refresh behavior) provide a better overview of the TDI servers monitored. There is also an indicator for whether AM is running or not.

Action Manager connection handling has been enhanced so that if AM loses connection to a TDI Server, then only the single API failure Trigger is invoked; all the others Triggers defined for that Server are suspended until connection is re-established.

One last big news item is that AMC now supports multiple versions of TDI Servers to be monitored (even simultaneously): v.6.1.1, v.6.1 and v.6.0. Of course, versions prior to v.6.1 will not provide newer functionality like Tombstones.

2.10 Secure Remote Commandline FC extended

The Remote Command Line component can now also connect to i5 and zOS servers, and is now based on the 2.2 version of the RXA library.

2.11 zLDAP now supported as password store for password synch.

The TDI password-catcher plugins can now be configured to store this information in the directory server running on z/OS.

2.12 Password Catcher Plug-in for Windows

The Password catcher for Windows/ActiveDirectory now provides a PKCS7 encapsulation of the password change event messages sent to TDI over MQe. In addition, there are now options that allow you to include (or exclude) specific LDAP suffixes, as well as groups (although this does not included nested groups).

2.13 SAX Parser

The SAX Parser has now been enhanced to work with or without group tags, allowing you to parse XML documents that contain multiple data element names/types.