TIBCO® Patterns

Installation

Version 6.1.2 | June 2024

Contents

Installation Overview

This section provides an overview of the installer.

Installers

The installers for the TIBCO® Patterns product are named:

TIB_tps_*n.n.n_platform*.zip

> **Note:** In the installer package name *n.n.n* is replaced with the release number and *platform* is replaced with a platform label indicating the machine and OS environment this package is for. To know the list of supported platforms, see the product *readme*.

In this document, the term TIBCO Patterns Server is used as a generic term for a single server executable for any of the TIBCO Patterns products.

For instructions on how to start and shut down the TIBCO Patterns server, see Running the TIBCO Patterns Server.

Installation Modes

Three installation modes are available: GUI, console, and silent.

GUI Mode

In the GUI mode, the installer presents screens that allow you to make choices about product selection, product location, and so on.

Console Mode

You can use the console mode to run the installer from the command prompt or terminal window. This is useful if your machine does not have a GUI environment.

Silent Mode

Silent mode installs the product using either default or custom settings that are saved in a response file. Silent mode installs the product without prompting you for information.

Installer Account

Microsoft Windows

You must have administrator privileges for the machine on which this product is installed. If you do not have administrator privileges, the installer exits. You must then log out of the system and log in as a user with the required privileges, or request your system administrator to assign the privileges to your account.

If you intend to install the product on a network drive, ensure that the account used for installation has permission to access the network drive.

UNIX

Any user can install this product.

- Regular (non-root) user
- Super-user (root).

While installing this product on UNIX platforms, ensure that the same user account is used to install all TIBCO Patterns products.

A graphic environment such as CDE or X Windows is required to run the installer in GUI mode.

Installer Log File

The installer log file, `tps_`*`version`*`_prodinfo` is written to the `TIBCO_HOME/_installInfo` folder. To change the location of the installer log file, specify the option `-V logFile="`*`myLogFile`*`"` when you run the installer.

The installer log file captures the following information:

- Installation environment details such as the installation location, operating system details, and so on.
- List of assemblies installed.

- Information related to the Ant scripts that are executed by the installer.

Installation Requirements

This section describes the disk space requirements, system memory requirements, software requirements, and supported platforms for this product.

Disk Space Requirements

Before installing this product, extract the contents of the installation archive to a temporary directory. The installer files consume up to 450 MB of disk space.

Temporary Disk Space Required by the Installer

The installer requires at least 575 MB of free space in the temporary directory. On Microsoft Windows, the temporary directory typically is:

`%SystemDrive%:\Documents and Settings\`*`user_name`*`\Local Settings\Temp`.

If your system does not have sufficient disk space in the default temporary area, you can run the installer with a different temporary directory by using the following option when starting the installer:

```
-is:tempdir /new_temp
```

where */new_temp* has sufficient free disk space.

The installer calculates the disk space required in the product home location for the selected components. The calculation is done before the actual installation (copying of files to the system) begins. The installer proceeds only if sufficient free disk space is available in the product home location.

However, if disk space is consumed by another process while the installer is copying the files, and if the required disk space is thereby reduced, the installer may fail and then display a failure message.

While performing installation, avoid running other processes that consume disk space in the product home location.

Disk Space After Installation

This product can consume 10 MB of free space under TIBCO_HOME for a minimal installation and 510 MB for a full installation.

System Memory Requirements

A minimum of 824 MB+ of physical memory is required.

Installation Modes

The following sections describe the installation process in the available installation modes:

- GUI Mode
- Console Mode
- Silent Mode

GUI Mode

1. Open the physical media or download the package.
2. Extract the contents of the package to a temporary directory.
3. Navigate to the temporary directory.
4. Run TIBCOUniversalInstaller (use the *.cmd* file for Windows, the *.bin* file for UNIX platforms). You can do so in one of the following ways:
   - Double-click the installer file.
   - On the command prompt, type the name of the installer file (including the absolute path if necessary). The installer defaults to GUI mode.
5. Click the **Next** button on the Welcome dialog.
6. Read through the license text when the License Agreement dialog appears, select the **I accept the terms of the license agreement** radio button**.**
7. Click the **Next** button. The TIBCO Installation Home dialog displays.

   An installation environment isolates product installations. A product installed into an installation environment does not access components in other installation environments. An installation environment consists of a name and directory. You can choose a new environment or an existing environment.

   - **Create a new TIBCO_HOME** To install the product into a new installation environment, specify the following properties:

     Directory The directory into which the product is installed. Type a path or click Browse to specify the path or accept the default location.

     The path cannot contain special characters such as "*", "#", "?", ">", "<", "%", "&", "$", "`" or "|". The path cannot be the same as the path of an existing environment.

Name Identifies the installation environment. The name cannot contain special characters such as "*", "?", ">", "<", ":", "|", "/", "\", or quotation marks(").

The name is appended to the name of Windows services created by the installer and is a component of the path to the product in the Windows Start > All Programs menu.

— **Use an existing TIBCO_HOME** To install the product into an existing installation environment, select the environment from the drop-down list. The Name and Directory fields are populated automatically and cannot be edited.

When installing multiple products into an existing environment, ensure that the products are compatible. If they are not compatible, the product installations may be corrupted.

8. Click the **Next** button. The **Installation Profile Selection** dialog displays.
9. Accept the default or select the **Customize Installation** checkbox and check the checkboxes next to the features to be installed on the right.
10. Click the **Next** button.
11. Verify the list of product features selected for install in the Pre-Install Summary dialog.
12. Click the **Install** button to start the installation process.
13. Click **Next** to complete the installation.
14. Review the information listed in the **Post-Install Summary** dialog.
15. Click **Finish**.

Console Mode

1. Open the physical media or download the package.
2. Extract the contents of the package to a temporary directory.
3. Using a console window, navigate to the temporary directory.
4. Launch the installer in console mode.

   On Windows: **`TIBCOUniversalInstaller-x86-64.exe -console`**

   On Linux: **`TIBCOUniversalInstaller-lnx-x86-64 -console`**
5. Complete the installation by responding to the console window prompts.

Silent Mode

In silent mode, the universal installer does not prompt for any inputs during installation. Instead, the inputs are read from a response file that can be provided as a command-line parameter. If no value is specified, the installer uses the default `TIBCOUniversalInstaller.silent` file.

The `TIBCOUniversalInstaller.silent` file is packaged in the directory that contains the universal installer. Edit the file with information for your environment before launching the silent installation. The file includes comments that describe the installation properties you can set. While you can use the `TIBCOUniversalInstaller.silent` file, it's good practice to copy the file to a different name and use that file for the silent install.

If errors occur during installation, they are listed in the installation log file located in the *User_Home*`/.TIBCO` directory, for example `C:\Documents and Settings\<`*user_name*`>\.TIBCO.`

1. Open the physical media or download the package.
2. Extract the contents of the package to a temporary directory.
3. Using a console window, navigate to the temporary directory.
4. Copy the `TIBCOUniversalInstaller.silent` file to a new file.
5. Using a text editor, open the copied file and update the install location and features to install as follows:
   - Update the install location. For example, update the directory as follows:

     ```
     <entry key="installationRoot">C:\Users\me\tibco</entry>"
     ```

   - Update *ENV_NAME*.

     For example, to install using the existing environment named "*TIBCO_HOME*", update *ENV_NAME* as follows:

     ```
     <entry key="createNewEnvironment">false</entry>
     <entry key="environmentName">TIBCO_HOME</entry>
     ```

   - Update features to install. For features that do not need to be installed, change their values to false.
6. Run the following commands:

   `TIBCOUniversalInstaller -silent -V responseFile="`*myfilename*`.silent"`

Uninstallation

This section describes how to uninstall this product in the GUI and the Console modes.

GUI Mode

1. Shut down all running TIBCO Patterns applications.
2. Navigate to TIBCO_HOME/tools/universal_installer and run TIBCOUniversalInstaller.
3. On the TIBCO Installation Manager page, perform the following steps:
   — Select the **Uninstall Products From Selected TIBCO Home Location** radio button.
   — Select the TIBCO_HOME location from the TIBCO Home Location drop-down list.
4. The Welcome dialog opens. Click the **Next** button.
5. Choose an uninstallation option. The wizard provides two uninstallation options:
   — **Custom Uninstall** You can select the products to be removed.
   — **Typical Uninstall** The universal uninstaller removes all the products in this TIBCO_HOME.
6. Click the **Next** button. If you selected the **Custom Uninstall (Select The Products To Be Removed)** radio button, select the checkboxes for products to uninstall, and then click the **Next** button.
7. Review the Pre-Uninstall Summary and click the **Uninstall** button to start the uninstallation process.
8. Review the Post-Uninstall Summary and click the **Finish** button to exit the uninstall wizard.

   The installer asks "Would you like to clean up the /tools directory for this environment".
9. Click **Yes** to delete the installed tools.

Console Mode

1. Using a command window, navigate to the TIBCO_HOME\tools\universal_installer directory.

2. Type the following command at the command prompt:

```
Windows: TIBCOUniversalInstaller-x86-64.exe -console
Linux: TIBCOUniversalInstaller-lnx-x86-64.bin -console
```

3. Complete the uninstallation by responding to the console window prompts.

Note: The silent mode is not available for uninstallation.

Running TIBCO Patterns

This section explains how to run TIBCO Patterns.

Running the TIBCO Patterns Server

The command syntax for the TIBCO Patterns server is as follows:

```
TIB_tps_server
[-p | --port plaintext-port:encrypted-port | plaintext-port]
[-g | --gateway gateway-configuration-file]
[-G | --temp-dir temp-storage-dir]
[-t | --threads max-threads]
[-R | --restore-dir restore-dir]
[-B | --durable-data]
[-A | --auto-restore]
[-a | --restore-from common-dir]
[-c | --con-log console-log]
[-l | --query-log querylog]
[-r | --pid pid-file]
[-H | --hawk service,network,daemon]
[-F | --prefilter prefilter]
[-m | --memory memory-cap]
[-L | --max-loads max-loads]
[-P | --persist socket-option]
[-T | --thes sub-limit]
[-I | --idle-tran idle-timeout]
[-J | --joins join-limits]
[-M | --model-info model-file]
[-u | --ip-in ip-mode]
[-U | --ip-out ip-mode]
```

[*-v* | *--verbose*]
[*-V* | *--version*]
[*-d* | *--debug*]
[*-D* | *--socket-log socket-log*]
[*-h* | *-?* | *--help*]
[*-C* | *--gpu DEFAULT* | *LIST* | *RECOMMENDED-ONLY* | *ALL* | *device specification*
[*-K* | *--private-key private-key-filename*]
[*-k* | *--public-key certificate-filename*]
[*-Z* | *--trusted-store trust-store*]
[*-E* | *--loadable-data-dir directory_name*]
[*ipaddr...*]

Note: The command line options are case-sensitive. Command line options can be specified using the short form or the long form. For example, the verbose option might be specified using -v or --verbose.

Data Persistence Options:

Option	Argument	Description
-p *--port*	*port-spec*	Listen on the *specified* ports. *Port-spec* is either *plaintext-port:encrypted-port*, or just a plaintext port number. One of these can be disabled by specifying *OFF* instead of a port number. Disabling both is not allowed. The default plaintext port is 5051. The default encrypted port is 8051. Note, this only specifies port numbers. Encrypted communications are disabled by default, they can be enabled by -k and -K options.
-g *--gateway*	*config-file*	Run as a gateway server using the specified gateway configuration file. Gateway servers and gateway configuration files are explained in the section, Gateway Servers and Clustering.

Option	Argument	Description
-P *--persist*	*socket-option*	Use persistent socket connections. socket-option is one of I = Incoming only, O = Outgoing only, B (default) = both, N = neither.
-u *--ip-in*	*ip-mode*	Set the incoming IP protocols the server accepts. *ip-mode* can contain one of the following protocols: **4** - IPv4 only **6** - IPv6 only **M** - Use both protocols. The default value is **M**.
-U *--ip-out*	*ip-mode*	Set the outgoing IP protocols the server uses. This controls what protocol this server uses when creating connections to a remote server. *ip-mode* can contain one of the following protocols: **4** - IPv4 only. The default value is '4', use only the IPv4 protocol for outgoing connections. **6** - IPv6 only **M** - Use both protocols.
ip-address...		Accept connections from the specified IP addresses as well as "localhost". By default, the server only accepts connections from "localhost" (127.0.0.1 or [::1] depending on the IP mode selected). The authentication list consists of localhost (127.0.0.1 or [::1] depending on the IP mode selected) along with any addresses given in this address list. Only hosts in the authentication list can connect to the server. Addresses in the list may be any host name that can be resolved to an IP address. If IPv4 is enabled (it is enabled unless -u 6 was specified), addresses in this list may also be valid IPv4 format IP addresses. Wild card and subnet mask lengths may be used. For example: 129.48.32.*

Option	Argument	Description
		192.168.*.* 129.48.34.0/24. Note that the entries may need protection from being expanded by the shell or command interpreter. If IPv6 is enabled (the default on all platforms except Windows platforms predating Windows Vista, and if -u 6 or -u M are specified), IPv6 format IP addresses may be given. The standard text formats as defined by RFC-5952 are supported, including subnet masks. As with IPv4 addresses, these must be protected from expansion by the shell or command interpreter.

Encrypted Communications Options:

Option	Argument	Description
-K *--private-key*	*filename*	Private key file-name. If a single file contains both the private key and the public key, then either -k or -K can be omitted. If both are omitted, encrypted communications are disabled.
-k *--public-key*	*filename*	Certificate file-name.
-Z *--trusted-store*	*trust-store*	SSL trust store file name or directory name. If this option is omitted, the system-default trust store is used. To specify a system certificate store on Windows, prefix the store name with *wincs:* The standard system stores are *ROOT*, *MY*, *SPC*, and *CA*. The default is *wincs:ROOT*.

Search Options:

Option	Argument	Description
-F *--prefilter*	*GIP* | *SORT* | *PSI*	Specify the default prefilter. If this argument is not given the default prefilter is GIP. The default prefilter can be overridden at the time the table is created with an explicit prefilter selection.
-T *--thes*	*sub-limit*	This specifies a limit on the number of different words or phrases in a query string to which thesaurus substitutions are applied. Substitutions beyond the limit are ignored. There can be a large performance penalty for very large numbers of substitutions, this limits that penalty. The default is 40, which is more than enough for most applications.
-J *--joins*	*join-limits*	Sets limits for joined searches. **Preferred format for *join-limits*:** [*single-parent-rec*]:[*all-parent-recs*]:[*max-querylets*] • *single-parent-rec* is the maximum number of child record combinations for a single parent record that is found by the GIP prefilter • *all-parent-recs* is the maximum number of such combinations for all parent records returned by the GIP prefilter • *max-querylets* is the maximum number of leaf querylets allowed in any query **Note:** All of the above three are optional, but the two colons are required. For example, if *single-parent-rec* and *all-parent-recs* are not applicable, and *max-querylets* value is 2000, the format is -J ::2000 **Alternative legacy format for *join-limits*:**

Option	Argument	Description
		- J *single-parent-rec*[:*all-parent-recs*] **Note:** If only the *single-parent-rec* limit is specified, then *all-parent-recs* is set to be equal to *single-parent-rec* and *max-querylets* is left at the default. If only the *single-parent-rec* is used in this format, then the colon must be omitted. The alternative legacy format provides backward compatibility for existing applications, but the preferred format is more flexible and allows setting the maximum number of querylets. For more information, see the “Single Parent Search” and “Matching Compound Records” sections in *TIBCO Patterns Concepts* Guide.
-C *--gpu*		Enables Graphics Processing Unit (GPU) harnessing for each table and query. For information about GIP-GPU, see the "Prefilters" section in *TIBCO Patterns Concepts Guide*. **Note:** -C is not supported on gateways, that is, starting a server with -g *config-file* -C is not a supported configuration. For information, see Gateway Servers and Clustering.
	LIST	Lists available GPU devices and exits without starting the server. It indicates whether it meets recommendations of TIBCO for each device. Using non-recommended devices can degrade application performance.
	RECOMMENDED-ONLY	Starts the server with recommended devices harnessed. Usable, but non-recommended devices are not harnessed and cannot be requested by applications.

Option	Argument	Description
	DEFAULT	Starts the server using GPU devices according to TIBCO recommendations. By default, recommended devices are harnessed. Usable, but non-recommended devices are harnessed only if requested by the application.
	ALL	Starts the server with all usable GPU devices harnessed.
	device-specification	Starts the server using specific GPU harnessing. *device-specification* is a comma separated list of one or more pairs of *device-id*,*flag*[, *device-id*,*flag*]* • *device-id*: Device IDs can be obtained by using -C *LIST*. • *flag*: The flag value can be `D` or `O`. — `D` for a default device. Default devices are harnessed to tables if not specified by the application. — `O` for an optional device. Optional devices are harnessed to tables only if requested by the application.

Data Persistence Options:

Option	Argument	Description
-R *--restore-dir*	*restore-dir*	Enable checkpoint and restore using the indicated directory. The directory must already exist. If this argument is not given, the checkpoint feature is disabled.
-A		Auto restore. Upon start up, restore all checkpointed tables in the restore directory. This is ignored if the -

Option	Argument	Description
--auto-restore		R option is not specified.
-a *--restore-from*	*common-dir*	Restore from a common directory. Upon start-up all checkpointed tables in the indicated directory are restored. Checkpointing is not enabled by this option, although it may be used with the -R and -A options.
-B *--durable-data*		Enable the Durable Data feature. This requires the -R option. Incompatible with the -a option.

Logging Options:

Option	Argument	Description
-c *--con-log*	*console-log*	Write error messages to the specified file. By default, the server sends messages to the system log on UNIX and the application event log on Microsoft Windows.
-l *--query-log*	*query-log*	Log incoming queries to the specified file.
-v *--verbose*		Log additional data to the console log. Each connection attempt is logged. If a connection attempt is rejected, the reason is logged. Do not use this option in production as the console log grows rapidly, until the file system is full.
-D *--socket-log*	*socket-log*	Logs all messages sent and received by the server to the indicated file. As it produces a massive file, use it only at the request of TIBCO Support.

Information Options:

Option	Argument	Description
-M *--model-info*	*model-file*	Display header data for the specified Learn Model file, and then exit.
-V *--version*		This causes the TIBCO Patterns server to print out its version information and exit.
-d *--debug*		Start in debug mode. The server does not detach itself from the controlling terminal, nor does it put itself into the background.
-h or -? *--help*		Help. Print a usage message and exit.

Other Server Configuration Options:

Option	Argument	Description
-G *--temp-dir*	*temp-storage-dir*	This directory is used by TIBCO Patterns for temporary files. The file system containing this directory must have sufficient free space to store any model files transmitted through the gateway.
-t *--threads*	*max-threads*	Spawn no more than the specified number of command processing threads. The default is 4.
-r *--pid*	*pid-file*	Write the server's process ID to the specified file. The file is deleted when the server terminates.
-H *-hawk*	"*service*, *network*, *daemon*"-	Enable the Hawk interface. If this argument is given, the server registers with TIBCO Patterns Hawk as a

Option	Argument	Description
		micro-agent, enabling TIBCO Hawk to monitor the state of the server. *service*, *network,* and *daemon* are the transport parameters as defined for TIBCO Hawk. For more information, see TIBCO Hawk Interface.
-m *--memory*	*memory-cap*	Set a limit for the total amount of space used for all database tables in the server. The space is given in KB. The minimum allowed size is 1024. When the total memory used for all database tables in the process reaches the limit specified, any attempt to add additional records or tables is disallowed. If this argument is not given, no check is made. If enough memory cannot be allocated for an additional record or table, the process terminates. Thus this argument can be used to protect the server from aborting due to insufficient memory resources or to cap its memory use on shared systems.
-L *--max-loads*	*max-loads*	Set the limit for the number of parallel loads in a multitable restore. By default, the Auto restore on start-up loads up to maxthreads and three tables in parallel to reduce load time where maxthreads is as defined in the -t option. During command processing, parallel loads in multi-table restores are turned off by default as they may interfere with other command processing. This can be controlled through this option. Setting max-loads to a positive value allows up to that number of tables to be loaded in parallel during command processing. Setting it to zero removes all restrictions and loads all tables in parallel. Setting it to a negative value disallows parallel loading in multitable restores during the command processing (same as the default behavior). For the initial auto-restore, the limit is always the maximum of maxthreads and three or max-loads.

Option	Argument	Description
-I *--idle-tran*	*idle-timeout*	Controls the auto-timeout of idle transactions. *Idle-timeout* has the following format: *idle-time*`[:`*action*`]` where *idle-time* specifies the time in seconds. A transaction must sit idle before the action is applied. An *action* can contain one of the following arguments: **A** – Abort the transaction **E** – Abort the transaction if it has any **E**rrors, commit if it is otherwise **C** – forcibly **C**ommit the transaction **N** – perform **N**o action, this turns off idle transaction monitoring. If this argument is not specified, idle transaction monitoring is turned off. If an action is not specified, it defaults to A.
-E *--loadable-data-dir*	*loadable-data directory*	Specifies the directory for server-side data loads. The server receives a notification from the application to load a server-side data file following which, only server-side data files are stored in this directory. The default directory path is `workingdirectory/loadable_data`. In case of incorrect `loadatable_data_dir`, TIBCO Patterns shows a WARN message. You can also provide relative path-. inside `loadable_data_dir`.

UNIX only: on startup, the server detaches itself from the controlling terminal and redirects all output to the console and log files as specified in the command line arguments. This is the standard behavior for background "`daemon`" processes on UNIX platforms. The -d option disables this behavior.

Data Persistence Directories

To use the Checkpoint/Restore feature, or the Durable Data feature, a separate directory must be allocated for each running TIBCO Patterns server. This directory is called the *Data Persistence directory*. It is also sometimes referred to as the *Checkpoint/Restore directory,* although this name is obsolete. The server must have permission to add and delete files in this directory. The

Checkpoint/Restore and Durable Data features store data in this directory. No two running TIBCO Patterns servers can share the same data-persistence directory. Also, no other process must attempt to modify this directory or its contents in any way. When using the directory for the first time, it must be empty. After this, the directory is managed by the TIBCO Patterns server.

> ⚠ **Warning:** The Checkpoint/Restore feature is deprecated in favor of the Durable Data feature.

In a typical production environment, there would be one TIBCO Patterns server per machine. A data-persistence directory is created on the machine. The TIBCO Patterns server must always run as the same user that owns the data-persistence directory. Typically, a special user ID is created for the TIBCO Patterns server and the user owns the data-persistence directory. The server is then started using the `-R directory` argument, where `directory` is the full path name to the created directory. The TIBCO Patterns server then takes over the responsibility for managing the contents of the directory, and no user action is needed. When a server starts, it performs the necessary cleanup of incomplete operations.

Selecting a Data-Persistence Directory

Consider the following when allocating a data-persistence or restore directory:

- Accessibility. The TIBCO Patterns servers must be able to access the directory from the home directory where the server was started.
- File Name Limits. The file system must support long file names.
- Space. The file system or partition on which the directory resides must have enough space to store the data. For the checkpoint/restore feature, files may take up to twice the size of the raw data. For the Durable-data feature, an additional 5x space is needed for ongoing record updates. So, if a CSV file dump of the table data takes about 1GB, the checkpoint file may take up to 2GB, and durable-data updates take up to 6GB. Ensure that enough space is available on the device as TIBCO Patterns servers do not check for the availability of space before writing out the data.
- File Size Restrictions. The checkpoint/restore feature stores the data for each table in a single file. Durable data splits the data for each table across several files, but it stores approximately three times as much of the total data. For large tables, the files created by either feature could exceed the maximum file size on some file systems. Ensure that the file system supports sufficiently large files to accommodate your tables.
- Network Traffic. If the directory is on a remote mounted file system (for example, NFS), reads and writes to the data-persistence directory could generate a very large amount of network traffic which could affect overall network performance.

- Speed. Directories on remote devices or slow devices can significantly impact checkpoint/restore and durable-data performance. A fast local device is recommended.
- Reliability. Do not depend on the TIBCO Patterns servers, even when using checkpoints or durable-data, as your primary secure data repository. However, if losing the data-persistence directory significantly impacts the availability of critical services, consider putting the data-persistence directory on a mirrored or RAID device.

⚠ **Warning:** Security. The data-persistence directory must be placed on an encrypted drive, and must not be accessible to unauthorized users.

- Each running TIBCO Patterns server must have its unique data-persistence directory.

Restoring Checkpoint Data at Server Start-up

Specifying the *-A (--auto-restore)* option in addition to the *-R (--restore-dir)* option instructs the TIBCO Patterns to restore checkpointed data at start-up. All checkpointed tables, thesauri, models, and character-maps are restored. If any checkpointed object fails to restore, the server logs an error and exits.

⚠ **Warning:** Auto-restoring large tables (over 10M records) can take considerable time. Applications may encounter a timeout error if they connect to TIBCO Patterns while it is auto-restoring large tables.

Sometimes, when running multiple TIBCO Patterns servers containing the same tables, it might be convenient to keep a single copy of the tables and have all TIBCO Patterns servers load from this shared copy by using the *-a (--restore-from)* option. It loads all the checkpointed tables from the named directory. This does not enable checkpointing in the shared directory; it is a read-only operation. The shared directory and all of its files must be readable by the TIBCO Patterns server. The shared directory must not contain incomplete checkpoints. The TIBCO Patterns server validates the contents of the directory, and exits if an incomplete checkpoint is found in the shared directory.

Checkpointing, auto-restore, and shared restore might be combined. Data in the shared directory is loaded first, followed by data from the data-persistence directory. This might override the shared data.

To manage multiple TIBCO Patterns servers with a common data set, designate one TIBCO Patterns server as the master. This server is always started first and has checkpoint and restore enabled on the directory (use the -R and -A options). When started, this master server cleans up any incomplete operations left in the directory.

Warning: If the master TIBCO Patterns server performs checkpoint operations while another server is restoring tables, it can cause the other server to fail and exit. This is especially true if a table is deleted.

Data-Persistence Directory Contents

The contents of the directory are managed by the TIBCO Patterns server. The files in the directory are briefly described in the following table:

File name	
LockFlag	Used to coordinate access to the directory.
*object-name-type-flag*O000	A checkpoint file for the object named **`object-name`**. This is a binary data file. It contains the complete information for the in-memory object needed to restore the object. The *type-flag* is a one character flag indicating the object type. The possible values for the *type-flag* are: **`D`** : a Data table **`T`** : a Thesaurus table **`C`** : a Character map **`M`** : a Learn Model An example for the Learn model is **`Model1`**.MO000
*object-name-type-flag*O*hhh*	A checkpoint file for a previous version of the object named **`object-name`**. The *type-flag* is as described in the previous entry. This exists only if the system fails during a checkpoint operation. The TIBCO Patterns server cleans this up on start up. This is the original version of the checkpoint file that is being modified by the open transaction with the short ID: *hhh*, where *hhh* is three hex digits. A transaction never has a short ID of '000' and therefore, this is always distinct from the current committed version that has the suffix *type-flag*O000. This file is deleted when the associated transaction is closed.

File name	
*object-name-type-flag*N*hhh*	A new, possibly incomplete, checkpoint file. The file was created by the open transaction with the short ID: *hhh*, where *hhh* is three hex digits. The file is removed when the transaction is closed.
*object-name-type-flag*R*hhh*	This indicates that the object with the name: *object-name* and the type indicated by the *type-flag* was renamed by the open transaction with the short ID: *hhh*, where *hhh* is three hex digits. This file is deleted when the associated transaction is closed.
hhh.ipt	A log of checkpoint restore actions performed by the transaction with the short ID: *hhh*, where *hhh* is three hex digits. This file is deleted when the associated transaction is closed.
OINT *ODEF* *OCONT*	Sub-directories used by the Durable-Data feature.
OINT/hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh.oint *OINT/hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh.tmp*	Contains naming and object-level transactional information about a data object.
ODEF/hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh.def *ODEF/hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh.tmp*	Contains the immutable content of a data object (for example, the field names of a table).
*OCONT/hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh.**	Contains mutable data of a data object (for example, table records).

Encrypting Network Communications

TIBCO Patterns supports the TLS and SSL standards for encrypting network communications.

TIBCO Patterns supports both self-signed certificates and certificates signed by a trusted Certificate Authority. However, the configuration for self-signed certificates is significantly more complex. Using self-signed certificates with the Java API or .NET API might require custom API programming.

Encrypting network communications incurs a performance penalty. In most use cases this penalty is very small.

Memory Sizing Estimates

To estimate the memory required to load a data-set into Patterns, a rough estimate of the raw data size is required. This could be a CSV file-size, a byte-count extracted from a SQL database, etc.

For a GIP-indexed table, 5x the raw data size is sufficient for most use-cases. For a SORT-indexed table, 3x the raw data size is sufficient for most use-cases. For a PSI-indexed table, 7x the raw data size is sufficient for most use-cases. These estimates include a margin for query memory.

Warning: Memory requirements vary by data-set.

Gateway Servers and Clustering

The TIBCO Patterns server supports clustering to distribute table data across hardware. A cluster consists of one or more node servers and a single gateway server. Clustering is primarily useful for handling very large tables. A cluster may have up to 64 nodes.

Applications address the gateway server, which then distribute data and workload to the nodes. Most applications might not address nodes directly. This is considered an administrative function.

For a cluster to support checkpoint / restore, each node must be started with the data-persistence directory specified.

For a cluster to support durable-data, each node must be started with a data-persistence directory specified (via -R option) and durable-data enabled (via -B option).

The host machine for the gateway server must be on the authentication list for each of the nodes. See the description of the IP address list in the section: Running the TIBCO Patterns Server.

When starting the servers in a cluster the outgoing IP mode of the gateway (-U option) must match the incoming IP mode of the nodes (-u option). If they do not match the gateway may not be able to communicate with the nodes.

Gateway servers are started with the -g option. This is incompatible with the -R, -A, and -a options for specifying a data-persistence directory or an auto-restore directory (gateway servers do not store tables, character maps, thesauri, or Learn Models; data is kept on the nodes).

The gateway must be started with maxthreads (-t option) equal to the sum of the maxthreads value for each of the nodes. The gateway only distributes the workload. It does not perform any administrative functions such as starting or stopping node servers. It is recommended to start the gateway after all nodes have been started.

Cluster Configuration Files

Cluster configuration files contain only ASCII text. Lines must be 200 characters or less. Empty lines are ignored. Lines containing a hash mark (#) or semicolon are truncated at that character (this allows for commenting). After truncation, leading and trailing whitespace is ignored.

In the following example, square brackets indicate optional items.

The file has two sections.

- First is the list of named nodes, containing the connection information. Each node has three required lines:

  ```
  NODE node-name
  HOST host-name or IP-address
  PORT port-number
  ```

  Node names must be unique.

- Second is the SSL flag. This flag controls whether communications between the gateway and nodes are encrypted.

  ```
  SSL ENABLED | DISABLED
  ```

SSL encryption defaults to disabled. If encryption is enabled, the gateway server must use the correct trust store (see the --trust-store option), and node servers must be launched with SSL configured (see the --port, --private-key, and --public-key options).

To use encryption, all nodes must have encryption enabled.

Note: Node names must contain only numbers (0-9), letters (a-z, A-Z), and underscores. Node names are case sensitive. Keywords on the other hand are case insensitive.

Sample Configuration File

```
NODE Alpha
           host 10.75.99.1
           port 7051
NODE Beta
           host 10.75.99.2
           port 7052
NODE Gamma
           host 10.75.99.3
           port 7053
NODE Omega
           host 10.75.99.3
           port 7053
SSL ENABLED
```

Changing a Configuration File

Changes to a configuration file do not take effect until the gateway is restarted or the configuration file is explicitly reloaded through a load cluster configuration command. See *TIBCO® Patterns Concepts Guide* for a description of the load cluster configuration command.

Monitoring a Cluster

It is recommended to monitor the health of a production cluster. If TIBCO Hawk is not in use, you can write a simple program that can periodically call the `cmdstats` function on the gateway and every node.

TIBCO Hawk Interface

The TIBCO® Patterns server can act as a Hawk micro-agent allowing it to be monitored by TIBCO Hawk. By default, the TIBCO Patterns server does not enable Hawk monitoring. To enable Hawk monitoring, the server must be started with the command-line option:

```
-H "service,network,daemon"
```

Where *service*, *network*, and *daemon* are the connection parameters for TIBCO Hawk as defined in the TIBCO Hawk documentation. For example,

```
-H "7474,,tcp:7474"
```

The above are the default settings. If a value is not given, the default value is used. If a `daemon` is not given, the second comma can be omitted. If both `network` and `daemon` are not given, both commas can be omitted. Thus, the following are all valid means of expressing the default values:

```
-H ""
-H ","
-H ",,"
```

Note that the argument is required. Giving no argument for the `-H` parameter is not allowed, even if the argument is to be empty quotes. If the TIBCO Rendezvous network is the local machine, use blank for the network, not localhost.

Only TIBCO Rendezvous transport is supported. If TIBCO Hawk enablement is requested, but it cannot establish a connection to the indicated TIBCO Rendezvous service, the server prints a warning message to the console log and continues with the startup. Startup does not fail if the TIBCO Hawk interface cannot be enabled. So make sure to check the console log to verify that TIBCO Hawk was enabled.

It is not necessary to install TIBCO Hawk or TIBCO Rendezvous on the machine where the TIBCO Patterns server is running, as long as the connection information specifies a reachable TIBCO Rendezvous service. If the TIBCO Rendezvous service is not available, TIBCO Patterns server tries to start a service on the local machine by starting the TIBCO Rendezvous *daemon*: "rvd". If this *daemon* is not defined in the PATH, TIBCO Hawk enablement fails with an error message in the console log and the server startup continues as described previously.

The TIBCO Patterns server is compatible with TIBCO Hawk and TIBCO Rendezvous . For information about the backward compatibility of micro agents with earlier versions of TIBCO Hawk, see the documentation of TIBCO Hawk and TIBCO Rendezvous.

Hawk Methods

The Hawk interface provides the following methods. Most of these methods are available as both subscription and invoke methods. All the subscription methods push data at fixed time intervals. Unless noted, they take a single argument that is the time interval in seconds. A minimum time of 10 seconds is enforced, times lower than 10 seconds are rounded up to 10 seconds. Other values are rounded to the next lower 5 second interval.

getServerInfo

Though the Subscription service is provided, the output of this method remains unchanged for the life of the TIBCO Patterns server process.

Output Parameters for getServerInfo

Parameter	Description
ProductName	TIBCO® Patterns Note that the final entry is reserved for special versions and is not currently used.
ServerVersion	The full release version string.
Prefilter	The prefilter enabled. Prefilter names are: GIP, SORT, PSI, NONE.
CheckpointDirectory	The data-persistence directory if checkpoint/restore or durable-data is enabled. Otherwise, it is `(not-enabled)`.
CheckpointAutoRestore	**True** if the server was started with the **Auto-Restore** option. Otherwise, it is **false**.
DurableData	**True** if the server was started with the **Durable Data** option. Otherwise, it is **false**.
ClusterConfiguration	The name of the configuration file for the gateway if it is a Gateway server. Otherwise, it is "`not a gateway`".
PlainTextPortNumber	The server listens on this port for plaintext communications.
EncryptedPortNumber	The server listens on this port for encrypted communications.
MaxThreads	The maximum number of command-processing threads that the server uses.
MemoryCap	If started with a fixed memory cap on table memory, the server shows the memory limit in KB; otherwise, it is zero.
TransactionTimeout	The duration (in seconds) for which the transaction must sit

Parameter	Description
	idle before it is automatically closed. If a transaction timeout period is not set, the value is returned as -1.
TransactionAction	The action performed on transactions that exceed the timeout period. This parameter can be in one of the following states: A – Abort the transaction E – Abort the transaction in case of errors, otherwise commit it. C – Forcibly commit the transaction N – No action is required.

StopServer

This takes no arguments and returns no values. It is an action method that shuts down the TIBCO Patterns server process. This is not available as a subscription.

PushTableInfo

This provides information on all loaded tables. Note that if there are no loaded tables, this report has one line with a TableName value of "...No Tables...".

Output Parameters for PushTableInfo

Parameter	Description
TableName	The name of the table.
TableType	The type of table. The possible values are “Standard”, “Child”, “Parent”, “Unknown”.
Prefilter	The name of the prefilter for this table: GIP, SORT, PSI, or NONE
CheckpointStatus	Indicates the checkpoint status. Either the time of the last checkpoint for this table or one of the following: • "`Checkpoints Disabled`" - if checkpointing is not enabled. • "`Never Checkpointed`" - if checkpointing is enabled but this

Parameter	Description
	table has not been checkpointed. • "`Automatic`" - if durable data is enabled.
NumRecords	The number of records currently in the table.
NumFields	The number of fields in the table.
KeyMemory	The number of KB of memory used to store the unique record key index.
RecordMemory	The number of KB of memory used to store record data.
HeaderMemory	The number of KB of memory used for various bookkeeping structures.
IndexMemory	The number of KB of memory used to store all indexes other than the unique record key index.
TotalMemory	The total number of KB of memory used to hold this table.
ParentName	The name of the parent table of this table, if this is not a "Child" table this value is: `*No Parent*`.
TableTranId	The full ID of the transaction currently holding this table. The value is `zero` if no transaction is holding this table.
TableTranState	The state flags for this object. The value is an `OR` of the following bits: **0x01** – the object exists, **0x02** the object is updated, **0x04** the object is held, **0x08** the object is added, **0x10** the object is deleted, **0x20** the object is renamed, **0x40** the object is replaced. For a gateway server the state may also include: **0x4000** the object has different states on different nodes. Deleted tables are not listed. So, the 'deleted' state is never actually returned.

PushThesaurusInfo

This provides information on all the loaded thesauri. Note that if there are no loaded thesauri, this report displays one line with a ThesaurusName value of "`...No Thesauri...`".:

Output Parameters PushThesaurusInfo

Parameter	Description
ThesaurusName	The name of the thesaurus.
ThesaurusType	The type of thesaurus, it can be one of: "substitution", "weighted term", or "combined"
NumTerms	The number of unique tokens in the thesaurus. It is useful when comparing two thesauri for equivalence. If this number is not the same, the thesauri are different. Note that this number is only an approximate indicator of the size of the thesaurus.
ThesaurusTranId	The full ID of the transaction currently holding this thesaurus. The value is zero if no transaction is holding this thesaurus.
ThesaurusTranState	The state flags for this object. The value is an `OR` of the following bits: **0x01** – the object exists, **0x02** the object is updated, **0x04** the object is held, **0x08** the object is added, **0x10** the object is deleted, **0x20** the object is renamed, **0x40** the object is replaced. For a gateway server the state may also include: **0x4000** the object has different states on different nodes. Deleted thesauri are not listed. So, the 'deleted' state is never actually returned.
CheckpointStatus	Indicates the checkpoint status. Either the time of the last checkpoint for this thesaurus or one of the following: • "`Checkpoints Disabled`" - if checkpointing is not enabled. • "`Never Checkpointed`" - if checkpointing is enabled but this table has not been checkpointed. • "`Automatic`" - if durable data is enabled.

pushModelInfo

This provides information on all loaded RLink Models.

Output Parameters for pushModelInfo

Parameter	Description
ModelName	The name of the RLink Model.
ModelMetadata	Metadata of the model.
NumFeatures	Number of features in the model.
ModelTranId	The full ID of the transaction currently holding this rlink model. The value is zero, if no transaction is holding this rlink model.
ModelTranState	The state flags for this object. The value is an `OR` of the following bits: **0x01** – the object exists, **0x02** the object is updated, **0x04** the object is held, **0x08** the object is added, **0x10** the object is deleted, **0x20** the object is renamed, **0x40** the object is replaced. For a gateway server the state may also include: **0x4000** the object has different states on different nodes. Deleted rlink models are not listed. So, the 'deleted' state is never actually returned.
CheckpointStatus	Indicates the checkpoint status. Either the time of the last checkpoint for this rlink model or one of the following: • "`Checkpoints Disabled`" - if checkpointing is not enabled. • "`Never Checkpointed`" - if checkpointing is enabled but this table has not been checkpointed. • "`Automatic`" - if durable data is enabled.

pushCharmapInfo

This provides information on all loaded character maps. The two preloaded character maps are:

- `=PUNCT=` Punctuation-sensitive map
- `=STD=` Standard default character map always appears in this report.

Output Parameters for pushCharmapInfo

Parameter	Description
CharacterMapName	The name of the character map.
CharacterMapTranId	The full ID of the transaction currently holding this character map. The value is zero if no transaction is holding this character map.
CharacterMapTranState	The state flags for this object. The value is an OR of the following bits: **0x01** – the object exists, **0x02** the object is updated, **0x04** the object is held, 0x08 the object is added, **0x10** the object is deleted, **0x20** the object is renamed, **0x40** the object is replaced. For a gateway server the state may also include: **0x4000** the object has different states on different nodes. Deleted character maps are not listed. So, the 'deleted' state is never actually returned.
CheckpointStatus	Indicates the checkpoint status. Either the time of the last checkpoint for this character map or one of the following: • `Checkpoints Disabled` - If checkpointing is not enabled. • `Never Checkpointed` - If checkpointing is enabled but this table has not been checkpointed. • `Automatic` - If durable data is enabled.

pushQueryInfo

This provides information on query performance. It takes no input arguments. The numbers reflect the performance of search queries, they do not include score records commands. It produces a summary report of query performance statistics for time periods of the last: 1 Minute, 5 Minutes, 15 Minutes, and since the server started or since the last `resetQueryInfo` method was invoked.

In the output parameters, all times are in seconds. If no queries were completed during the time period, the time values are -1 and counts are 0.

Output Parameters for pushQueryInfo

Parameter	Description
StatsWindow	Indicates the time period for the statistics on this line.
QueryCount	The number of query commands completed in the time period.
MaxQueryTime	The elapsed time for the longest running query completed in the time period.
MinQueryTime	The elapsed time for the shortest running query completed in the time period.
AvgQueryTime	The average elapsed time for all the queries completed in the time period.
AvgQueryOverhead	The average amount of time taken for parse query arguments, wait for an available command processing thread, and transmit query results.
QueryThroughput	The average number of queries per second processed during this time interval. This is computed as QueryCount and its duration.
QueryMaxThroughput	This is the maximum throughput in queries per second over any one minute interval during the time report period or the last 17 minutes, whichever is less.
QueryConcurrency	This is a rough approximation for the average number of command-processing threads in concurrent use for the query processing during the report time period.

resetQueryInfo

This is an action method, it takes no argument and returns no values. It resets all query statistics.

commitTransaction

This forcibly commits a transaction. It takes one argument.

Input Parameter for commitTransaction

Parameter	Description
TranId	The full transaction ID of the transaction to be committed. This is a 64-bit integer value.

abortTransaction

This aborts a transaction. It takes one argument.

Input Parameter for abortTransaction

Parameter	Description
TranId	The full transaction ID of the transaction to be committed. This is an integer value.

pushTransactionInfo

This provides information on active transactions. This method takes one argument in addition to the report interval.

Input Parameter for pushTransactionInfo

Parameter	Description
TransactionIdleTime	This is a time in seconds. Only transactions that have been idle longer than this time are included in the report. If this time is less than or equal to zero all active transactions are reported.

Output Parameters for pushTransactionInfo

Parameter	Description
TranId	The full transaction ID for the transaction.
TranType	One of “implicit” or “explicit”.
TranIsGateway	A true or false value. `True` if this is a gateway transaction.

Parameter	Description
TranIsRunning	A true or false value. `True` if there was a command in progress on this transaction.
TranIdleTime	The time in seconds the transaction has been idle. This is the time since the last command executing under this transaction completed.
TranErrorCount	This is the number of commands executed under this transaction that failed.
TranLogSize	This gives the number of items in the transaction rollback log. It is an estimate of the amount of work performed under this transaction.

If no transactions are meeting the input criteria, a single line is returned with all integer values set to zero, all true and false values set to `false`, and the TranType parameter set to: “No-Transactions”.

Server Threads

TIBCO Patterns servers are multithreaded. If your host machine has multiple processors, a TIBCO Patterns server process uses them in parallel. Even on a single-processor host, it can do database loads and dumps in parallel with queries.

The thread usage of TIBCO Patterns servers depends on the platform. To avoid resource contention, the number of request handler threads is limited to four by default. You can adjust this limit with the -t option on the command line. The optimal number of threads depends on the number of CPUs available, the hardware and memory configuration, the volume and configuration of data loaded into the server, the expected query rate, and the type of queries. In the recommended configuration of a single TIBCO Patterns server running on a dedicated machine, the number of threads might generally be equal to the number of virtual CPUs available on the machine. At least one thread is required.

Warning: Do not start a server with a single thread if it is used as a gateway for multitable searches.

IPC Timeouts

The timeout for inter process reads and writes for TIBCO Patterns servers is fixed at 30 seconds. If a read or write request to a client process is inactive for 30 seconds, the server closes the socket. Additionally, if a persistent connection is not used for 90 seconds, the server closes the socket.

There are no timeouts on a read from a client using the "C", Java, or .NET interfaces to a TIBCO Patterns server. The read request waits until the server responds, the connection is dropped, or the read request is interrupted by a signal.

Running in a Container

You can containerize TIBCO® Patterns and run it in a Docker or Kubernetes environment. To containerize TIBCO Patterns, you must build and run the Docker images using the bundled Docker ZIP file.

A container consists of an entire runtime environment: an application; all its dependencies, and configuration files needed to run it bundled into one package. You need not worry about the differences in operating system distribution in the case of container applications. For information about Docker concepts, such as Docker file, Docker Image, and Container, see Docker documentation.

The Dockerfiles are delivered as a ZIP file on the TIBCO eDelivery website. Download the TIB_tps_x.x.x_container.zip file and extract its content to a separate directory.
To build the Docker image for:

- Patterns, see the ReadMe file located under the directory `build/patterns/ReadMe.txt`
- Patterns Gateway, see the ReadMe file located under the directory `build/gateway/ReadMe.txt`

To deploy using:

- helm, see the ReadMe file located under the directory `k8s_helm/ReadMe.txt`
- k8s_deployments, see the DeploymentInstructions.txt file located under the directory `k8s_deployment/DeploymentInstructions.txt`

Shutting down the TIBCO Patterns Server

The TIBCO Patterns server can be shut down in three ways:

- An authorized client may connect and issue the `svrshutdown` command.

- UNIX only: You may send the signal SIGTERM to the server. The UNIX kill command sends SIGTERM by default.
- Windows only: if TIBCO Patterns is run as a Windows Service, all the standard service-control tools can be used.
- If TIBCO Patterns is run interactively in debug mode (-d option), it can be stopped using standard keyboard interrupts (for example, control-C).
- Windows only: the `taskkill.exe` command can be used to do an immediate, unclean shutdown.
- If TIBCO Hawk is enabled, the TIBCO Hawk administrator can shut down the server through TIBCO Hawk interface.
- The NSC sample program provides a `svrshutdown` command that can be used to shut down the server and wait for the shutdown to complete.

When a shutdown is initiated, the server first stops accepting new commands, then waits for all currently running commands to complete, frees all data, removes the PID file if one was created (see -r option), and then exits. Long running commands check for a termination request and exit early, but the shutdown process may take a considerable amount of time, especially if the server has very large tables.

If a shutdown is taking too long or appears to be hung, sending the TIBCO Patterns server a SIGTERM signal during the shutdown process forces the server to do an immediate, unclean shutdown. This is not recommended as it may not properly clean up all client socket connections, potentially causing problems in the client processes, and may not clean up the PID file (see -r option).

It is recommended that the -r option be used when starting the TIBCO Patterns server. As the PID file is removed as the last step in the shutdown process, it can be used as a flag for when the shutdown is complete. If you attempt to start a new TIBCO Patterns server before the first server has completed shutdown the start of the new server may fail. The NSC sample program `svrshutdown` command can take advantage of the PID file to both wait for the server to complete shutdown, and, if it is taking too long, force shutdown with a SIGTERM signal. This is the recommended shutdown procedure.

Running a TIBCO Patterns Server on Microsoft Windows

The following section provides information to run TIBCO Patterns Server on the Windows platform.

Running a TIBCO Patterns Server as a Microsoft Windows Service

To run a TIBCO Patterns server as a service, use the sc program to install the service. An example command is as follows:

```
sc create "TIBCO Patterns Server" binpath= "c:\path_to_executable\TIB_tps_server.exe"
```

Note the space after `binpath=` . Use the correct name for the TIBCO Patterns executable and make sure to specify the correct path to the TIBCO Patterns executable. To remove the service, type:

```
sc delete "TIBCO Patterns Server"
```

After the service has been installed, you can use the Services applet to control the service called "TIBCO Patterns Server." This applet is in the **Control Panel** under **Administrative Tools and System and Security => Administrative Tools**. Use it to start and stop the service, set it to start automatically on reboot, and specify command-line options.

Note that when the service is set to start automatically, any command-line options must be included in the binpath option when installing the service.

For example,

```
binpath= "c:\path_to_executable\TIB_tps_server.exe -p 5054 -c TIB_tps.con"
```

You cannot set autostart parameters from the Windows applet.

Also, there is an Event Viewer applet in the **Control Panel** under **Administrative Tools**. The TIBCO Patterns servers logs start, stop and error messages in the Event Log. Use the Event Viewer applet to view them.

You can use the `-c` option to write the error messages to a file instead of the Event Log. Command-line errors still appears in the Event Log.

Note: You cannot use the `-M`, `-?` or `-h` options when you run the TIBCO Patterns server as a service.

Running a TIBCO Patterns Server on UNIX Variants

File Descriptor Limits

To reduce connection overhead, the TIBCO Patterns server persists socket connections. It can use up to 1024, or one-third of the file descriptors available to the process, whichever is lower, to persist the incoming socket connections. Additionally, it uses the same number of file descriptors to persist the outgoing socket connections to other TIBCO Patterns servers.

When the Data-Durability feature is enabled, the server uses several dozen file descriptors per table and a strict minimum number of descriptors are required to start the server. The server does not start without at least 2048 descriptors.

Beyond these limits, the server uses transient connections. File descriptor limits should be set (using the `ulimit` command) to at least 1024, with 64K (65536) or higher being preferred.

Update the ulimit to use the Patterns server with the data-durability feature.

Using an Alternate Memory Allocator

The TIBCO Patterns server supports using alternate memory allocators; tcmalloc™ (4.4.5 or later) and jemalloc (3.6.0 or later). Depending on your application workload, an alternate allocator might reduce the TIBCO Patterns memory footprint by up to 60%. Alternate memory allocators can also affect command latency and throughput, either positively or negatively, depending on the application workload. These effects are usually, but not always, insignificant.

Note: You must always test an alternate memory allocator for its effect on performance, before using it in a production environment. The *jemalloc* and *tcmalloc* can be installed on Redhat™ 7 or Redhat™ 8.

Depending on your Linux variant, perform the following installation steps:

jemalloc

1. Subscribe to the Extra Packages for Enterprise Linux (EPEL) repository.

2. Run sudo yum install jemalloc.

The library should appear as `usr/lib64/libjemalloc.so.1.`

tcmalloc

1. Subscribe to the Extra Packages for Enterprise Linux (EPEL) repository.

2. Run sudo yum install gperftools.

The library should appear as `usr/lib64/libtcmalloc.so`.

Usage with TIBCO Patterns Patterns

Perform the following steps to use an alternate memory allocator with TIBCO Patterns:

1. Locate the library of the alternate memory allocator.

Alternate Memory Allocator	Library
jemalloc	`usr/lib64/libjemalloc.so.1`
tcmalloc	`usr/lib64/libtcmalloc.so`

2. Identify the command line used to launch the TIBCO Patterns server.

3. Modify the command line used to launch the TIBCO Patterns server: prepend the `LD_PRELOAD` environment variable.

Examples

Original command line:

```
TIB_tps_server your-options...
```

With the jemalloc allocator:

```
LD_PRELOAD=/usr/lib64/libjemalloc.so.1  TIB_tps_server your-options...
```

With the tcmalloc allocator:

```
LD_PRELOAD=/usr/lib64/libtcmalloc.so  TIB_tps_server your-options...
```

Running the TIBCO Patterns Java Embedded Server

The TIBCO Patterns Java Embedded Server is delivered as a relocatable library. The library is delivered in the same bin directory as the standard server executable.

The library name is platform-specific.

For Windows:

```
TIB_tps_embedded.dll
```

For Linux:

```
libTIB_tps_embedded.so
```

This library is platform specific. Only the Linux and Windows platforms are supported. The library is also built to work with a 64-bit Java Virtual Machine (Java command). The library cannot be used with a 32-bit JVM.

The library may be copied to any convenient location for use in your application. To use the embedded server, the application must specify the full path to this relocatable library in the initialization call. For details, see the `NetricsEmbedded` and `NetricsEmbeddedSettings` classes in the Java API documentation.

Server Application Interface

The application interface to the TIBCO Patterns server is explained in *TIBCO Patterns Programmer's Guide* and the documentation contained in the Java or .NET application interface packages.

Library Dependencies

This section provides a list of shared object libraries or dynamically linked libraries used by the TIBCO Patterns server broken down by platform. All of these libraries are standard libraries that exist on any properly installed and configured system for the given platform. As the list of libraries needed is under the control of the platform vendor, this list may change any time.

Linux x86 - 64-bit

This should be compatible with most Intel or AMD x86 compatible hardware platforms running most versions of Linux. Specifically, the TIBCO Patterns server is built for RedHat Linux.

- `linux-vdso.so.1`: Virtual dynamic shared object library
- `librt.so.1`: Time functions
- `libpthread.so.0`: POSIX threads

- `libuuid.so.1`: Universally Unique Identifier library
- `libdl.so.2`: Dynamic loader routines
- `libm.so.6`: Basic math functions
- `libgcc_s.so.1`: GCC run time routines
- `libc.so.6`: Standard “C” library
- `ld-linux-x86-64.so.2`: Dynamic linker/loader

Microsoft Windows x86 - 64-bit

The TIBCO Patterns Server is built for Intel or AMD x86 compatible 64-bit processors and the Microsoft Windows 64-bit Operating System. For the list of versions, see the *Readme* file.

The libraries listed here are those that the TIBCO Patterns server calls directly. The functions within these DLLs often link to other Microsoft Windows DLLs. It is not possible to determine exactly which DLLs are needed for any particular application. A properly installed Microsoft Windows OS always provides the necessary supporting DLLs. Consult Microsoft Windows support for list of the DLLs required for your particular machine.

- `ADVAPI32.dll`: Basic "C" library routines
- `WS2_32.dll`: Microsoft Windows Sockets 2 interface
- `KERNEL32.dll`: Basic Microsoft Windows Kernel operations
- `CRYPT32.dll`: Encryption facilities
- `USER32.dll`: Windows user interface
- `bcrypt.dll`: Encryption facilities

Upgrading an TIBCO® Patterns Installation

This section provides guidelines for upgrading an TIBCO® Patterns installation. These guidelines provide only a starting point for migrations and upgrades, you must review each installation for relevant details.

Pre-Migration Steps

Complete the following steps to upgrade the TIBCO® Patterns installation:

1. Review the functionality of the new and existing versions
2. Install the new version of TIBCO® Patterns
3. Transfer data from the existing installation to the new installation
4. Update applications that use TIBCO Patterns

Warning: Before upgrading production systems, it is a good practice to perform an upgrade on a test system.

Reviewing Functionality

With each release of TIBCO® Patterns, features are modified. Usually new features are added, but occasionally existing features are deprecated or deprecated features are removed.

Features are always deprecated for at least one release cycle before they are removed. Therefore, when updating TIBCO® Patterns from one release version to the next, there should be no immediate compatibility issues.

Perform the following tasks when reviewing the functionality of new and existing functions:

- Review all applications for the features that they use. This includes the following features:
    - Server Features (e.g., checkpoint/restore)
    - APIs and Plug-ins (Java, .NET, BW6, etc.)
    - Query functionality
- Review the *TIBCO® Patterns Release Notes* for new and existing versions of TIBCO® Patterns, and all versions in-between. All feature changes are documented in the *TIBCO® Patterns Release Notes*.

Note: Reviewing *TIBCO® Patterns Release Notes* is critical if the upgrade skips versions.

Installing the New Version

To install the new version of TIBCO® Patterns, obtain the installer for the new version and run the installer.

You can run the new installation on the existing hardware, or new hardware. If existing and new installations are run on the same hardware simultaneously:

- There may be a temporary loss of performance while both installations are running.
- Configuration must avoid conflicts between the installations. This includes:
    - TCP ports
    - Log locations
    - Data-persistence locations (i.e. checkpoint/restore folders)

Transferring Data

You can transfer data from the existing installation to the new installation in the following ways:

- Performing a fresh load of data fresh from the original data source.
- Copying the Checkpoint files.

  This method is only available when checkpoint/restore is in use on the existing installation.

  Complete the following steps to copy the checkpoint files:
    - Perform a checkpoint-all operation on the existing installation.
    - Copy the checkpoint files to the checkpoint existing of the new installation.
    - Perform a restore-all operation on the new installation.
- Using durable data
  This method is only available when the existing installation is 5.7.1 or a newer version.
  With the durable-data feature, there is effectively no transfer of data. The new installation can be run using the existing durable-data directory.
  Complete the following steps to use the existing durable-data directory:
    - Shut down the existing installation.
    - Back up the durable-data directory.
    - Run the new installation using the existing durable-data directory.

Note:

- Shut down or pause applications that perform write operations while performing the data transfer.
- If the existing installation is version 5.4 or earlier, the copying checkpoint files method only transfers tables. Thesauri, character-maps, and RLINK models must be transferred before the restore-all operation is performed.

Updating Applications

Complete the following steps to update the applications that use TIBCO® Patterns:

1. Update the following APIs and Plug-ins:
    a. Update the Java applications to use the API jar from the new version.
    b. Update the .NET applications to use the API dll from the new version.
    c. Update the BW5 applications to use the matching plug-in.
    d. Update the BW6 applications to use the matching plug-in.
2. Change the applications that use a removed feature.
3. Change the applications that use deprecated features.

Generally, the Java API, .NET APIs, BW6 plug-in, and BW5 plug-in are forward compatible. Updating them is often sufficient to update an application. The primary exception is when an application uses deprecated methods, and those methods are removed.

When updating an application, check if any deprecated methods are in use. Changing these to non-deprecated methods is recommended. The continued use of deprecated methods may cause issues during later upgrades.

For more information about deprecated and removed features, see *TIBCO® Patterns Release Notes*.

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

Documentation for TIBCO® Patterns is available on the TIBCO® Patterns Product Documentation page.

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to TIBCO Community.

Legal and Third-Party Notices

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

TIBCO, the TIBCO logo, the TIBCO O logo, ActiveMatrix BusinessWorks, BusinessConnect, TIBCO Hawk, TIBCO Rendezvous, TIBCO Administrator, TIBCO BusinessEvents, TIBCO Designer, and TIBCO Runtime Agent are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT (S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.