HP Performance Insight

Process Insight Report Pack

Software Version: 1.0

User Guide

Document Release Date: October 2007
Software Release Date: October 2007

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

Java™ is a US trademark of Sun Microsystems, Inc.

Oracle® is a US registered trademark of Oracle Corporation, Redwood City, California.

Microsoft® is a US registered trademark of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

Windows® and MS Windows® are US registered trademarks of Microsoft Corporation.

Documentation Updates

This guide's title page contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://ovweb.external.hp.com/lpe/doc_serv/

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP Software Support web site at:

http://www.hp.com/go/hpsoftwaresupport

HP Software Support Online provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

For more information about HP Passport, go to:

http://h20229.www2.hp.com/passport-registration.html

Contents

1 Overview

HP Business Process Insight (HPBPI) provides you with visibility into the health and performance of the business processes. HPBPI allows you to assess the financial and business impact of delays or blockages in a process due to an IT performance problem or other incident such as an IT outage. HPBPI is also able to record business process metrics such as the time between two steps in a process. For example, you can configure HPBPI to notify you when an order takes more than one day to complete or when there is a delay with one of your Gold customer's orders. HPBPI provides a Web-based Business Process Dashboard which allows the user to see the current state of their business and to see the scale of any business impact that may be occurring.

HP Performance Insight (PI) is a performance management and reporting application. Long-term data collection, in-depth analysis, and automated web-based reporting are this application's primary strengths.

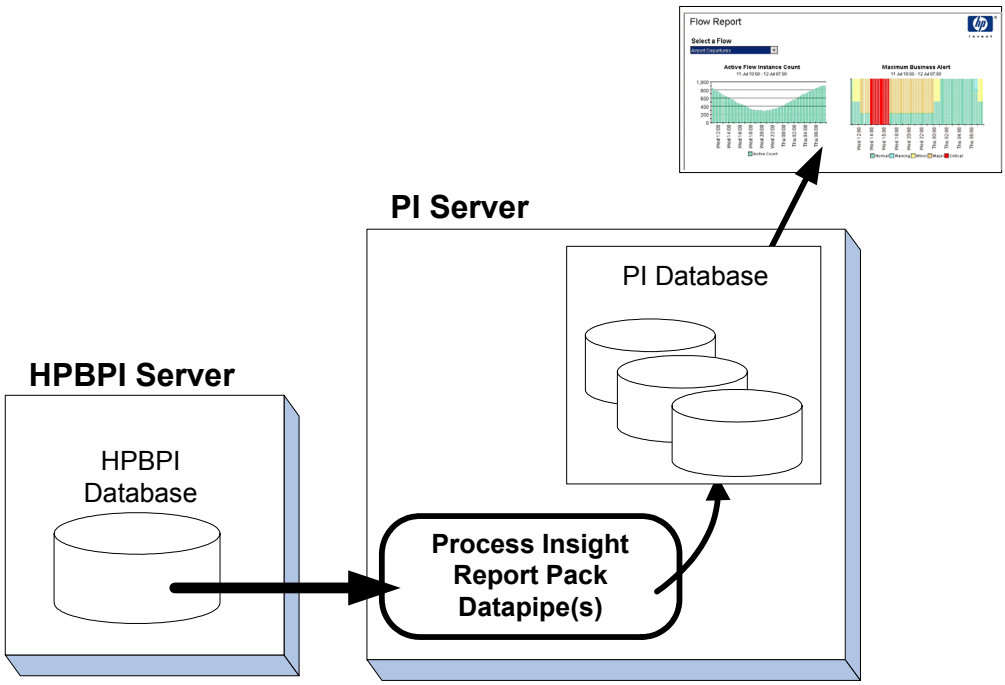
By providing a Report Pack for HPBPI you can now maintain historical reporting for HPBPI. This report pack is called the Process Insight Report Pack.

High-Level Architecture

The Process Insight Report Pack installs on PI. A set of datapipes are scheduled to run on the PI server, and these datapipes bring data across from the HPBPI database into the PI database. You can then choose from a variety of out-of-the-box reports to view the performance of your underlying HPBPI system. These reports provide both a near real time view of your HPBPI system as well as historical performance over time.

The high-level architecture of the Process Insight Report Pack is shown in Figure 1.

Figure 1 High-Level Architecture



Package Contents

The Process Insight Report Pack provides the following components:

- The HPBPI Scraper
- A set of PI Database tables
- A set of reports

The HPBPI Scraper

The HPBPI Scraper is a Java application that periodically retrieves data from the HPBPI database and brings this data across to your PI database.

The HPBPI Scraper maintains a set of datapipes to handle the retrieval of the HPBPI data. There is a datapipe for each type of data that the Scraper brings across from HPBPI.

More information about the HPBPI Scraper and the different datapipes can be found in Chapter 3, The HPBPI Scraper.

A Set of PI Database Tables

There is a set of PI database tables specifically for holding the data that comes from HPBPI. There are tables to hold information about HPBPI flows, flow instances, node instances, business process metrics, etc..

The list of PI database tables is described in PI Base Table Structures on page 43.

A Set of Reports

The HPBPI Report Pack provides a set of out-of-the-box reports.

There are reports for the following areas of HPBPI:

- Flows
- Nodes
- Thresholds
- Metric Alerts
- Metric Statistics
- Metric Values

For each of the above areas there are reports that show near real time views of the data, as well as providing hourly, daily, weekly and monthly summary reports.

The reports are described in more detail in Chapter 4, Reports.

2 Installation

The HPBPI Report Pack installs using the standard PI package manager installation utility. There are however some post-installation tasks required on both the HPBPI server and the PI server.

Let's look at the steps involved to install the HPBPI Report Pack.

Prerequisites

The HPBPI Report Pack requires the following software versions:

- HPBPI Version 2.20 or later
- PI Version 5.3 or later

Installing the HPBPI Report Pack

The steps to installing the HPBPI Report Pack are as follows:

- Task 1: Stop OVPI Timer and extract the package from the report pack CD
- Task 2: Install the HPBPI Report Pack
- Task 3: Install the HPBPI Database Indexes and other post-install tasks
- Task 4: Configure the HPBPI Report Pack
- Task 5: Re-Start the OVPI Timer Service

Task 1: Stop OVPI Timer and extract the package from the report pack CD

1. Log in to the system. On UNIX systems, log in as `root`.
2. If the `OVPI Timer` process is running, stop it and wait for processes to terminate.

   Windows:

   a. From the Control Panel, select `Administrative Tools > Services`

   b. Select `OVPI Timer` from the list of services.

   c. From the Action menu, select `Stop`.

   UNIX:

   As root, type one of the following:

   — HP-UX: `sh /sbin/init.d/ovpi_timer stop`

   — Sun: `sh /etc/init.d/ovpi_timer stop`
3. Insert the report pack CD in the CD-ROM drive.

   Windows: A menu displays automatically.

   UNIX:

   a. Mount the CD (if the CD does not mount automatically).

   b. Navigate to the top level directory on the CD.

   c. Run `./setup`
4. Type `1` in the choice field and press `Enter`.

   The install script displays a percentage complete bar. When the copy is complete, the install script starts Package Manager. The Package Manager welcome window opens.

Task 2: Install the HPBPI Report Pack

1. When the Package Manager welcome window appears, click `Next`.
2. The Package Location window opens:

   a. Click to select the option: `Install`

   b. Accept the default destination folder or browse to a different directory if necessary.

   c. Click `Next`.

3. The Report Deployment window opens:
   a. Type your user name and password for the PI Application Server.
   b. Click `Next`.
4. The Package Selection window opens:
   a. Click to select the check box next to:

      Process_Insight

      The `Process_Insight` report pack includes the HPBPI Scraper/datapipe.
   b. Click `Next`.
5. The Type Discovery window opens:
   a. Un-check the option to run the Type Discover

      That is, you do not run Type Discover as it is not required.
   b. Click `Next`.
6. The Selection Summary window opens:
   a. Click `Install`.

   The Installation Progress window opens and the install begins.
7. When the install finishes, a package installation complete message appears.
   a. Click `Done`.

Task 3: Install the HPBPI Database Indexes and other post-install tasks

The HPBPI Report Pack requires that some of the data tables within the HPBPI database are configured with additional indexes. You also need to ensure that any of your HPBPI custom metrics are upgraded to use Version 2 of the custom metric mechanism (as introduced with HPBPI Version2.20).

Please refer to Post Install Tasks on the HPBPI Server on page 17 for details of the post installation tasks you now need to carry out on your HPBPI server.

➤ Do not proceed with the installation until you have carried out the Post Install Tasks on the HPBPI Server on page 17

Task 4: Configure the HPBPI Report Pack

Before the HPBPI Report Pack can be used, there are some post-installation tasks that you need to carry out. These tasks include things such as specifying the details of the HPBPI database to which your HPBPI Report Pack will be connecting. Please refer to Post Install Tasks On the PI Server on page 19 for details of the post installation tasks you now need to carry out on your PI server.

➤ Do not proceed with the installation until you have carried out the Post Install Tasks On the PI Server on page 19

Task 5: Re-Start the OVPI Timer Service

1. Log in to the system. On UNIX systems, log in as `root`.
2. Start the `OVPI Timer` process.

   Windows:

   a. From the Control Panel, select `Administrative Tools > Services`

   b. Select `OVPI Timer` from the list of services.

   c. From the Action menu, select `Start`.

   UNIX:

   As root, type one of the following:

   — HP-UX: `sh /sbin/init.d/ovpi_timer start`

   — Sun: `sh /etc/init.d/ovpi_timer start`

The HPBPI Report Pack is now installed and running.

Post Install Tasks on the HPBPI Server

On the HPBPI server, you need to do the following:

- Install database indexes
- Upgrade custom metrics
- Check Instance cleaner configuration

Install Database Indexes

The HPBPI Report Pack uses a Java application (the HPBPI Scraper) to periodically retrieve information from the HPBPI database and copy this information into the PI database. This periodic retrieval of information from the HPBPI database is referred to as a “scrape”.

The HPBPI Scraper achieves this scraping of information by issuing SQL select statements against various tables within the HPBPI database. These SQL select statements select new information that has occurred since the last scrape. In other words, the select statements select their data based on date/time columns within each of the HPBPI tables. For these select statements to be handled efficiently by the underlying database, you need to set up some additional indexing on the HPBPI data tables involved. The HPBPI Report Pack provides a script for setting up the necessary indexing.

There are two script files provided depending on whether the HPBPI database is MSSQL or Oracle. The two script files are called:

```
BPI_PI_Indexes.oracle.sql
BPI_PI_Indexes.mssql.sql
```

These files are located under the *DPIPE_HOME*/packages directory, in the sub-directory:

```
Process_Insight/Process_Insight.ap/bpi_dpipe
```

where *DPIPE_HOME* is the directory in which PI is installed.

To install the necessary HPBPI database indexes you need to do the following:

1. Copy the appropriate BPI_PI_Indexes SQL file to your HPBPI server.
2. Run the BPI_PI_Indexes SQL file against the HPBPI database.

   This creates the necessary indexes required for the HPBPI Report Pack.

Upgrade Custom Metrics

HPBPI Version 2.20 introduced a new version (version 2) of the custom metric mechanism. This mechanism was added so that custom metrics could use the millisecond time stamp information that came from the HPBPI Metric Engine.

If your HPBPI Report Pack is going to retrieve data from an HPBPI server that has custom metrics, then you must ensure that all these custom metrics are upgraded to use version 2 of the custom metric mechanism. If your HPBPI server has custom metrics that are using version 1 of the custom metric mechanism then the metric values for these custom metrics are not retrieved by the HPBPI Scraper and are therefore not shown in the PI Reports. Metric statistics and metric alerts are handled correctly, however metric values are ignored.

All custom metrics need to be upgraded to ensure that they call the `METRIC_SEND_EVENT_V2` stored procedure and that the custom metric definition within the `METRIC_CustomTypes` table defines the `ParameterVersion` attribute to be the value `2`.

Check Instance Cleaner Configuration

The HPBPI Scraper issues select statements against the various tables within the HPBPI database to retrieve instance data for completed flow instances, node instances and metric instances. For the Scraper to be able to retrieve these completed instance records you need to ensure that the HPBPI instance cleaners (for both the Impact Engine and the Metric Engine) are leaving completed instances in the HPBPI database long enough for the HPBPI Scraper to retrieve them.

If the instance cleaners do remove these completed instances before the Scraper has retrieved them, then the HPBPI Report Pack simply does not see these records and they do not appear in any of the reports.

Post Install Tasks On the PI Server

Once the HPBPI Report Pack is installed on your PI server you need to configure the connection to the HPBPI database. Without the database connection details the HPBPI Scraper cannot retrieve any data from HPBPI.

The HPBPI Scraper has many configuration options and these are all described in Chapter 3, The HPBPI Scraper. However, there is only one set of configuration options that is required, and that is the database connection details. There are however a couple of optional settings that you might wish to review at this time, so they are mentioned here.

The HPBPI Scraper reads its configuration details from the following file:

```
DPIPE_HOME/lib/bpi_dpipe.properties
```

where *DPIPE_HOME* is the directory in which PI is installed.

When you make changes to the `bpi_dpipe.properties` file, this affects the next invocation of the HPBPI Scraper.

The configuration options described here are as follows:

- HPBPI Database connection (required)
- Epoch start date (optional)
- Logging (optional)
- Test Configuration Mode (optional)

HPBPI Database Connection

The entries in the `bpi_dpipe.properties` file that set the HPBPI database connection details are as follows:

```
dbDriver
dbProtocol
dbUser
dbPasswordEncoding
dbPassword
```

Let's go through and look at each property:

- `dbDriver`

  You specify the JDBC driver that the HPBPI Scraper is to use.

  The HPBPI Scraper ships with JDBC JAR files for both MSSQL (2000 and 2005) and Oracle (9i and 10g) and so example setting are as follows:

  MSSQL:

  ```
  dbDriver = com.inet.tds.TdsDriver
  ```

  Oracle:

  ```
  dbDriver = oracle.jdbc.driver.OracleDriver
  ```

- `dbProtocol`

  You specify the jdbc protocol for the connection as a URL.

  This URL includes the host name and the port number of the HPBPI database. For Oracle, the URL also incudes the SID of the HPBPI database.

  Example setting are as follows:

  MSSQL:

  ```
  dbProtocol = jdbc:inetdae7://hostname:1433
  ```

  Oracle:

  ```
  dbProtocol = jdbc:oracle:thin:@hostname:1521:oraSid
  ```

- `dbUser`

  You specify the user name that the HPBPI Scraper is to use when it connects to the HPBPI database. This user name needs to be the user name that has access to the HPBPI database.

  For example:

  ```
  dbUser = hpbpiuser
  ```

- `dbPasswordEncoding`

  You specify whether the password text you supply for the `dbPassword` property is in plain text or encoded text form.

  For example:

  ```
  dbPasswordEncoding = false
  ```

  If this property is not specified then it defaults to `true` and your database password needs to be entered in encoded form.

  Refer to Encoding the Database Password on page 21 for details of how to generate an encoded password.

- `dbPassword`

  You specify the password for the user specified by the `dbUser` property.

  For example:

  ```
  dbPassword = hpbpi
  ```

  If you do not wish to have your password displayed in plain text within this properties file, then you can choose to encode the password. Either set the property `dbPasswordEncoding` to `true` or comment it out, and then follow the details outlined in Encoding the Database Password on page 21 to produce an encoded password string.

Encoding the Database Password

The HPBPI Scraper provides an encoder that allows you to produce an encoded version of your password which you can then use in the `bpi_dpipe.properties` file.

The encoder is located in the directory:

```
DPIPE_HOME/lib/bpi_dpipe
```

where *DPIPE_HOME* is the directory in which PI is installed.

The actual file name of the encoder depends on whether you are running PI on UNIX or Windows:

For Windows, the file name is `encodePasswd.bat`

For UNIX, the file name is: `encodePasswd`

The `encodePasswd` script requires the following two environment variables to be set:

- `JAVA_HOME`

  This needs to point to your installed Java home directory.

- `DPIPE_HOME`

  This needs to be set to your installed PI home directory.

The `encodePasswd` script takes one parameter, which is your database password string. The script then outputs the encoded version of that password to the screen. You can then cut and paste the encoded version of the password into your `bpi_dpipe.properties` file.

For example:

If your database password is `myGreatPassword`, you can produce the encoded version of this password by starting a command window and running the `encodePasswd` script as follows:

```
$ ./lib/bpi_dpipe/encodePasswd myGreatPassword
1;4h4u0m173y3e0v1h6345330f253m2t4w
```

The encoded version of the password is:

```
1;4h4u0m173y3e0v1h6345330f253m2t4w.
```

You would then set the properties within your `bpi_dpipe.properties` file as follows:

```
dbPasswordEncoding = true
dbPassword = 1;4h4u0m173y3e0v1h6345330f253m2t4w
```

Example Database Connection Properties

Here are some example data connection property settings:

Oracle Database

```
dbDriver   = oracle.jdbc.driver.OracleDriver
dbProtocol = jdbc:oracle:thin:@Server1.hp.com:1521:oraSid
dbUser     = hpbpiuser
dbPassword = hpbpi
dbPasswordEncoding = false
```

Oracle Database – with Database Encoding

```
dbDriver   = oracle.jdbc.driver.OracleDriver
dbProtocol = jdbc:oracle:thin:@Server1.hp.com:1521:oraSid
dbUser     = hpbpiuser
dbPassword = 1;4h4u0m173y3e0v1h6345330f253m2t4w
```

MSSQL Database

```
dbDriver    = com.inet.tds.TdsDriver
dbProtocol  = jdbc:inetdae7://Server2.hp.com:1433
dbUser      = hpbpiuser
dbPassword  = hpbpi
dbPasswordEncoding = false
```

MSSQL Database – with Database Encoding

```
dbDriver    = com.inet.tds.TdsDriver
dbProtocol  = jdbc:inetdae7://Server2.hp.com:1433
dbUser      = hpbpiuser
dbPassword  = 1;4h4u0m173y3e0v1h6345330f253m2t4w
```

Epoch Start Date

The HPBPI Scraper runs in one of two modes:

- definitions

  In this mode, the HPBPI Scraper retrieves the current status information for all flows, nodes and thresholds.

  The Scraper simply retrieves the status information at the time the Scraper is run. For example, if a flow's status is `Blocked` then the Scraper retrieves the fact that the flow is `Blocked`.

- instances

  In this mode, the HPBPI Scraper retrieves all flow/node/metric instances that have completed since the last time the HPBPI Scraper scraped the database.

  So in this mode, the Scraper needs to keep track of the last time it scraped for data. When the Scraper runs, it asks for all instance data that is between the last time the Scraper ran and now.

  What happens when the Scraper runs for the very first time? What does it specify as the "last time" it ran?

By default, when the HPBPI Scraper runs in `instances` mode for the first time, it uses the current time as the "last time". In other words, the Scraper asks for all data since now. This means that, by default, the first run of the Scraper in `instances` mode returns no data. Subsequent runs of the Scraper retrieve the instance data as it occurs.

If you want the very first run of the Scraper to start from a specific date and time other than the current time, then you can specific the configuration property `epochStartTimestamp`.

If you want to configure a starting date/time for your HPBPI Scraper then refer to Epoch Start Date on page 58 for more details.

Logging

By default, the HPBPI Scraper logs all output to log files. The name of the log file depends on the mode that the Scraper is running in.

The log files are located in the directory:

DPIPE_HOME`/log`

where *DPIPE_HOME* is the directory in which PI is installed.

The names of the log files are:

- `bpi_dpipe_definitions.log`

  This holds the log output for all runs of the HPBPI Scraper in `definitions` mode.

- `bpi_dpipe_instances.log`

  This holds the log output for all runs of the HPBPI Scraper in `instances` mode.

Each time the HPBPI Scraper runs, it appends its output to an existing log file.

If you want to change the logging settings then refer to Logging on page 23 for more details.

Test Configuration Mode

If you have specified your database connection details incorrectly, then when the HPBPI Scraper comes to run it will be unable to connect to the HPBPI database and log this as an error to the log file. So it is important that you check the log file to ensure that your Scraper is able to run successfully.

If you want to manually test your HPBPI Scraper and verify that the database connection details are correct, then you can. This involves the setting of the property `testConfigAndExit` and then running the HPBPI Scraper standalone.

If you want to see how to run the HPBPI Scraper in test mode then refer to Test Configuration Mode on page 58 for more details.

Installed File Locations

The files for the HPBPI Report Pack all install under sub-directories of the PI install directory (referred to as *DPIPE_HOME*).

Let's look at each sub-directory and the HPBPI Report Pack files that are placed within them:

- *DPIPE_HOME*/bin

  This directory holds the executable that invokes the HPBPI Scraper.

  The file name depends on the installation platform.

  For a UNIX installation the executable is called `bpi_dpipe`. For a Windows installation the executable is called `bpi_dpipe.exe`.

- *DPIPE_HOME*/data/startup

  This directory holds the file `bpi_dpipe.ini`.

  This .ini file configures the *DPIPE_HOME*/bin/bpi_dpipe executable.

  The `bpi_dpipe.ini` file specifies the Java class of the HPBPI Scraper to be invoked, the Java classpath to be used, and some Java properties to be passed into the Scraper class.

- *DPIPE_HOME*/data/ImportData/Process_Insight

  This directory is created when you install the HPBPI Report Pack.

  This directory is where the HPBPI Scraper outputs its data as CSF (Character Separated Format) files ready for the PI utility `ee_collect` to process and place into PI data tables.

- *DPIPE_HOME*/lib

  This directory holds the following files:

  — `bpi_dpipe.properties`

    This is the main property file for the HPBPI Scraper. It contains all the property settings for the HPBPI Scraper, such as the HPBPI database connection details.

  — `bpi_dpipe_history.properties`

    This file is created the first time the HPBPI Scraper runs in `instances` mode. It is then used to hold the latest set of timestamps for subsequent invocations of the Scraper. You should not edit or remove this file.

  — `trendtimer.sched`

    This file has entries in it to schedule the times that the HPBPI Scraper is run, as well as when the various trends are to be run.

  — `BPIDP_collect_*.teel`

    These files specify how the PI utility `ee_collect` processes the CSF files that are created by the HPBPI Scraper.

- *DPIPE_HOME*/lib/bpi_dpipe

  This sub-directory contains the following files:

  — bpi_dpipe.jar

  This JAR file contains the code for the HPBPI Scraper.

  — Opta.jar

  This JAR file contains the JDBC client libraries for accessing an MSSQL database. This JAR file contains the libraries necessary for accessing MSSQL 2000 and 2005.

  The data/startup/bpi_dpipe.ini file includes this JAR on the classpath of the HPBPI Scraper.

  — ojdbc14.jar

  This JAR file contains the JDBC client libraries for accessing an Oracle database. This JAR file contains the libraries necessary for accessing Oracle 9i and 10g.

  The data/startup/bpi_dpipe.ini file includes this JAR on the classpath of the HPBPI Scraper.

  — encodePasswd (.bat)

  This script is used when you wish to produce an encoded database password. Refer to Encoding the Database Password on page 21 for more details of how to use this script.

- *DPIPE_HOME*/log

  By default, the HPBPI Scraper is configured to output logging information to the following files:

  — bpi_dpipe_definitions.log

  Log output is appended to this file when the Scraper is running in definitions mode. If the file does not exist then the Scraper creates it when required.

  — bpi_dpipe_instances.log

  Log output is appended to this file when the Scraper is running in instances mode. If the file does not exist then the Scraper creates it when required.

- *DPIPE_HOME*/scripts

  This directory contains the following files:

  — BPIDP__collect_definitions.pro

  This file runs the HPBPI Scraper in definitions mode and then runs the necessary ee_collects to process the data and place it into the PI data tables.

  An entry in trendtimer.sched invokes this .pro file at configured intervals.

  — BPIDP__collect_instances.pro

  This file runs the HPBPI Scraper in instances mode and then runs the necessary ee_collects to process the data and place it into the PI data tables.

  An entry in trendtimer.sched invokes this .pro file at configured intervals.

— `BPIDP__trend_<hourly/daily/weekly/monthly>.pro`

These files execute the set of `.sum` files necessary to summarize the data that has come from HPBPI into time periods, such as hourly, daily, weekly and monthly.

Entries in `trendtimer.sched` invoke these `.pro` files at configured intervals.

— `BPIDP_*.sum`

These files are the specifications for how the data that has come from HPBPI is to be summarized into the hourly, daily, weekly and monthly time periods.

These `.sum` files are invoked by the `BPIDP__trend_*.pro` files.

- *DPIPE_HOME*`/scripts/Oracle` or *DPIPE_HOME*`/scripts/Sybase`

It depends on your PI installation as to which of these sub-directories you will have on your system.

These directories contain the Oracle and Sybase versions of the supportive SQL procedures used by the HPBPI Report Pack.

The files are:

— `BPIDP_K_table_*.sql`

The SQL script `BPIDP_K_table_fix.sql` is executed after the HPBPI Scraper has been run in `instances` mode, and `ee_collect` has moved all the data into the PI tables. This script calls the SQL procedure `BPI_UpdateKTables` (defined in the file `BPIDP_K_table_update_proc.sql`). This SQL procedure looks at the latest set of data retrieved from HPBPI for all Nodes, Metrics and Thresholds, and ensures that the non-key columns in the PI property tables are updated to match these latest values.

Table 1 on page 27 lists the property tables that are checked and the columns that are updated.

The `BPI_UpdateKTables` procedure ensures that the non-key columns in the property tables are kept up to date as the new data comes across from HPBPI. For example, if the HPBPI developer changes the measurement period of a metric within his/her HPBPI system, this new measurement period is then correctly reflected in the `K_BPI_METRIC` table within PI.

Table 1 Property Tables and Non-Key Columns

Property Table Name	Non-Key Columns
K_BPI_NODES	k_node, k_x_pos, k_y_pos
K_BPI_METRICS	k_groupName, k_valueUnits, k_measurementPeriod
K_BPI_THRESHOLDS	k_thresholdType, k_thresholdColumnName, k_thresholdTest, k_alertCriticalLevel, k_alertMajorLevel, k_alertMinorLevel, k_alertWarningLevel, k_alertCriticalLevelDispUnits, k_alertMajorLevelDispUnits, k_alertMinorLevelDispUnits, k_alertWarningLevelDispUnits

— `BPIDP_setup_deadlineRangeTable.sql`

This SQL procedure is run when the HPBPI Report Pack is installed. This procedure populates the table `GB_BPI_DEADLINERANGES` with a range of values. These values are used within the flow instance deadline reports to allow the user to select lower and upper bounds for the flow instances that are displayed.

— `BPIDP_view_*.sql`

These SQL procedures create SQL views that are used by some of the reports within the HPBPI Report Pack.

- *`DPIPE_HOME`*`/reports`

The reports that make up the HPBPI Report Pack are held under this directory as follows:

— `Process_Insight`

The reports are held under this sub-directory.

— `deploy/system/Process_Insight`

The deployed versions of the reports are held under this sub-directory.

On-Going Maintenance

When the HPBPI Report Pack is installed and running, all log messages are logged, by default, to the following two files:

- *DPIPE_HOME*/log/bpi_dpipe_definitions.log
- *DPIPE_HOME*/log/bpi_dpipe_instances.log

where *DPIPE_HOME* is the directory in which PI is installed.

Each execution of the HPBPI Scraper appends new log messages to these files, so these files will grow without bound.

To prevent these files from growing too large and filling your disk, you should consider one of the following options:

- Periodically moving these files to an archive area.

  When the HPBPI Scraper is not running you can simply move these files to another folder for archiving. For example, you could schedule this to occur at the end of every day and then keep the previous week's worth of log files on disc.

  If the HPBPI Scraper runs and the log files do not exist in the DPIPE_HOME/log directory, the Scraper recreates them as required.

- Setting the log level of the HPBPI Scraper to ERROR.

  Running the Scraper with ERROR log level means that no log messages occur unless the Scraper encounters an error. This means that a normal run of the Scraper produces no log output. Running the Scraper with ERROR log level means that the log files should not grow unbounded.

  You would only consider switching the log level to ERROR once you had run the Scraper a few times and were confident that everything was running OK.

Package Removal

To remove the HPBPI Report Pack you need to do the following steps:

- Remove the HPBPI Report Pack from the PI server.
- Remove the PI-specific database indexes from the HPBPI database.

On the PI Server

Here are the steps required to remove the HPBPI Report Pack and datapipe from your PI system:

1. Log in to the system. On UNIX systems, log in as `root`.
2. If the `OVPI Timer` process is running, stop it and wait for processes to terminate.

   Windows:

   a. From the Control Panel, select `Administrative Tools > Services`

   b. Select `OVPI Timer` from the list of services.

   c. From the Action menu, select `Stop`.

   UNIX:

   As root, type one of the following:

   — HP-UX: `sh /sbin/init.d/ovpi_timer stop`

   — Sun: `sh /etc/init.d/ovpi_timer stop`
3. Run the Package Manager, and when the Package Manager welcome window appears, click `Next`.
4. The Package Location window opens:

   a. Click to select the option: `Uninstall`

   b. Click `Next`.
5. The Report Undeployment window opens:

   a. Type your user name and password for the PI Application Server.

   b. Click `Next`.
6. The Package Selection window opens:

   a. Click to select the check box next to:

   Process_Insight

   b. Click `Next`.
7. The Summary window opens:

   a. Click `Uninstall`.

   The progress window opens and the uninstall begins.
8. When the uninstall finishes, a completion message appears.

   a. Click `Done`.

The HPBPI Report Pack (including the HPBPI Scraper/datapipe) is now removed from your PI system.

On the HPBPI Server

The HPBPI Report Pack provides a script for removing the PI-specific indexing from the HPBPI database.

There are two script files provided depending on whether the HPBPI database is MSSQL or Oracle. The two script files are called:

```
BPI_PI_Indexes_Remove.oracle.sql
BPI_PI_Indexes_Remove.mssql.sql
```

These files are located under the *DPIPE_HOME*/packages directory, in the sub-directory:

```
Process_Insight/Process_Insight.ap/bpi_dpipe
```

where *DPIPE_HOME* is the directory in which PI is installed.

To remove the PI-specific indexes from the HPBPI database indexes you need to do the following:

1. Copy the appropriate BPI_PI_Indexes_Remove SQL file to your HPBPI server.
2. Run the BPI_PI_Indexes_Remove SQL file against the HPBPI database.

   This removes the PI-specific indexes from the HPBPI database.

3 The HPBPI Scraper

The HPBPI Report Pack includes the HPBPI Scraper.

This chapter looks at how the HPBPI Scraper works, and the datapipes it uses. There are also details of the configuration options available for the HPBPI Scraper.

Data Retrieval and Datapipes

The HPBPI Report Pack uses the following eight datapipes to bring data across from the HPBPI database:

- BPIDP_FLOWS

  This datapipe brings across the status of all non-deleted flow definitions.

  This includes data such as the current active flow count, the current IT impact status and the current highest metric alert for each flow.

- BPIDP_NODES

  This datapipe brings across the status of all node definitions for non-deleted flows.

  This includes data such as the current active node count, the current active weight and the current IT impact status for each node.

- BPIDP_THESHOLDS

  This datapipe brings across the status of all threshold definitions for non-deleted metrics.

- BPIDP_FLOWINST

  This datapipe brings across all completed flow instances for non-deleted flows.

  This includes the basic flow instance details such as the weight and duration of the instance, as well as the associated business data for this instance.

- BPIDP_NODEINST

  This datapipe brings across all completed node instances for non-deleted flows.

  This includes data such as the node instance start time, complete time and duration.

- BPIDP_METRICVALUES

  This datapipe brings across all completed metric values for non-deleted metrics.

  This includes data such as the value of each metric, the start time and completed time. If the metric is a deadline metric then the data also includes details such as the deadline date and the metric's performance against this deadline.

- BPIDP_METRICSTATS

  This datapipe brings across all completed metric statistics for non-deleted metrics.

  This includes data such as the total weight, total metric instance count and throughput details for each HPBPI metric collection interval.

- BPIDP_METRICALERTS

  This datapipe brings across all raised metric alerts for non-deleted metrics.

  This includes data such as the alert level, the time the alert was raised and the value of the metric instance that raised the alert.

The HPBPI Scraper Java application provides the necessary code to access the HPBPI database and retrieve the data for each of the above eight datapipes. However, rather than invoking the HPBPI Scraper for each datapipe, the Scraper combines the eight datapipes into two groups:

- definitions

  When the HPBPI Scraper is run to collect definitions, it connects to the HPBPI database and retrieves three sets of data. The Scraper retrieves the data for the three datapipes:

  — BPIDP_FLOWS

  — BPIDP_NODES

  — BPIDP_THESHOLDS

- instances

  When the HPBPI Scraper is run to collect instances, it connects to the HPBPI database and retrieves five sets of data. The Scraper retrieves the data for the five datapipes:

  — BPIDP_FLOWINST

  — BPIDP_NODEINST

  — BPIDP_METRICVALUES

  — BPIDP_METRICSTATS

  — BPIDP_METRICALERTS

The datapipes are grouped together like this because of the similarity of the information that is retrieved. The `BPIDP_FLOWS`, `BPIDP_NODES` and `BPIDP_THRESHOLDS` datapipes are all retrieving status information for each defined flow, node or threshold. This status information is the kind of information you would want to retrieve fairly often as this provides you with a near real time view of your HPBPI system. Also, the volume of data being retrieved each time is fairly small.

The “instance” based datapipes are all about retrieving data on each instance of a flow, node, metric value, metric statistic and metric alert. The number of instances to be retrieved with each scrape could be very large and so, unlike the definitions, you may choose to run an instance scrape less often.

By default, the HPBPI Scraper is scheduled to run a definitions scrape every 5 minutes and an instances scrape every 30 minutes. You can alter these according to your own requirements by editing the PI scheduler.

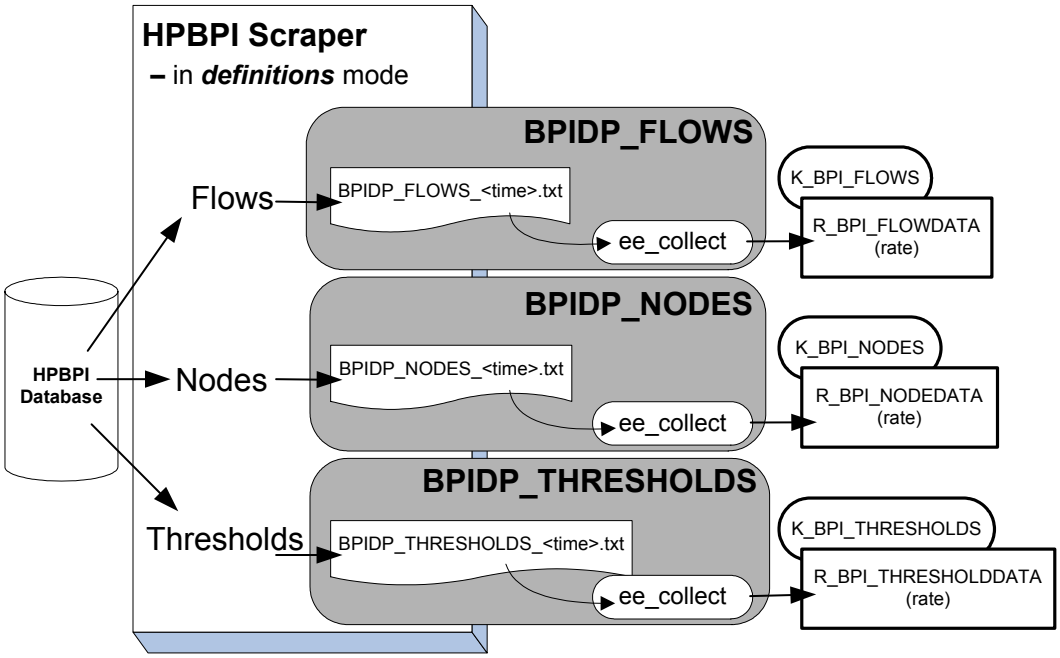
Definitions

When the HPBPI Scraper is run in `definitions` mode it accesses the HPBPI database and retrieves the necessary data for the three datapipes:

- BPIDP_FLOWS
- BPIDP_NODES
- BPIDP_THESHOLDS

For each datapipe, the HPBPI Scraper retrieves the necessary data from the HPBPI database and writes it to a new CSF file. Each datapipe then uses the PI utility `ee_collect` to process its respective CSF file and place the data into the PI database, as shown in Figure 2.

Figure 2 HPBPI Scraper in Definitions Mode



The running of the HPBPI Scraper in definitions mode, followed by the collection of the three datapipes (as shown in Figure 2) is configured within the file:

DPIPE_HOME`/scripts/BPIDP__collect_definitions.pro`

where *DPIPE_HOME* is the directory in which PI is installed.

Let's look at the three types of data that are retrieved when the HPBPI Scraper runs in definitions mode, as shown in Figure 2 on page 34.

Flows

When retrieving flow details, the HPBPI Scraper accesses the following HPBPI database tables:

- `flows`

  The flows table provides the list of non-deleted flows. For each of these flows, the Scraper retrieves data such as the flow name, the active current flow count and the current IT status impact.

  This data provides a snapshot of the current status of each flow.

- `metric_dim_metrics` and `metric_dim_thresholds`

  For each flow, the Scraper then uses these two metric tables to find out the current highest metric alert (if any) for the flow.

- `flow_instance`

  For each flow, the Scraper counts the number of flow instances that are in the states `Blocked`, `At Risk` and `Healthy`.

  For a large HPBPI database, with many active flow instances, this particular part of the HPBPI Scraper could take some time and affect the performance of your HPBPI database. For this reason, the HPBPI Scraper can be configured not to carry out this retrieval of the flow_instance table. Refer to Flow Instance Active/AtRisk/Blocked Counts on page 66 for more details.

The HPBPI Scraper writes this flow data to a CSF file. The CSF file is located in the directory:

DPIPE_HOME/data/ImportData/Process_Insight

where *DPIPE_HOME* is the directory in which PI is installed.

The name of the CSF file has the form:

BPIDP_FLOWS_*timestamp*.txt

where *timestamp* is the time that the HPBPI Scraper created this CSF file, with the time expressed as the number of milliseconds since 1970.

The `BPIDP_FLOWS` datapipe then runs `ee_collect` to process this CSF file and place the data into the PI database tables as shown in Figure 2 on page 34.

Nodes

When retrieving node details, the HPBPI Scraper accesses the following HPBPI database tables:

- `flows`

  The flows table provides the list of non-deleted flows.

- `nodes`

  The nodes table provides basic node definition data such as the node name and type, as well as the current active node count, the current node throughput and the status of any current IT impact.

  This data provides a snapshot of the current status of each node.

The HPBPI Scraper writes this node data to a CSF file. The CSF file is located in the directory:

DPIPE_HOME/data/ImportData/Process_Insight

where *DPIPE_HOME* is the directory in which PI is installed.

The name of the CSF file has the form:

BPIDP_NODES_*timestamp*.txt

where *timestamp* is the time that the HPBPI Scraper created this CSF file, with the time expressed as the number of milliseconds since 1970.

The BPIDP_NODES datapipe then runs ee_collect to process this CSF file and place the data into the PI database tables as shown in Figure 2 on page 34.

Thresholds

When retrieving threshold details, the HPBPI Scraper accesses the following HPBPI database tables:

- flows and metric_dim_metrics

  These tables provide the list of non-deleted metrics for non-deleted flows.

- metric_dim_thresholds

  The metric_dim_thresholds table provides basic threshold definition data such as the threshold name and type, as well as the current alert level for each threshold.

  This data provides a snapshot of the current status of each threshold.

The HPBPI Scraper writes the threshold data to a CSF file. The CSF file is located in the directory:

```
DPIPE_HOME/data/ImportData/Process_Insight
```

where *DPIPE_HOME* is the directory in which PI is installed.

The name of the CSF file has the form:

```
BPIDP_THRESHOLDS_timestamp.txt
```

where *timestamp* is the time that the HPBPI Scraper created this CSF file, with the time expressed as the number of milliseconds since 1970.

The BPIDP_THRESHOLDS datapipe then runs ee_collect to process this CSF file and place the data into the PI database tables as shown in Figure 2 on page 34.

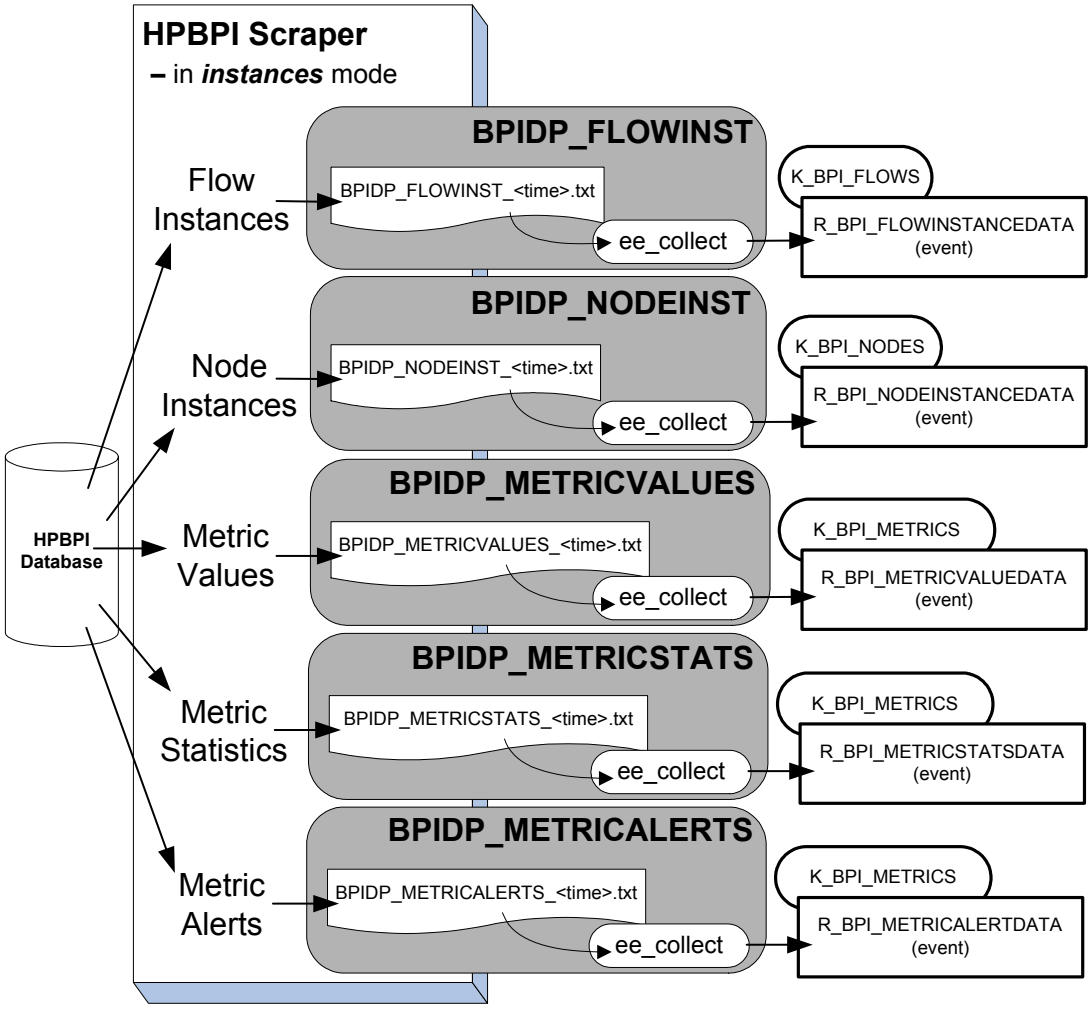
Instances

When the HPBPI Scraper is run in `instances` mode it accesses the HPBPI database and retrieves the necessary data for the five datapipes:

- BPIDP_FLOWINST
- BPIDP_NODEINST
- BPIDP_METRICVALUES
- BPIDP_METRICSTATS
- BPIDP_METRICALERTS

For each datapipe, the HPBPI Scraper retrieves the necessary data from the HPBPI database and writes it to a new CSF file. Each datapipe then uses the PI utility `ee_collect` to process its respective CSF file and place the data into the PI database, as shown in Figure 3 on page 37.

Figure 3 HPBPI Scraper in Instances Mode



The running of the HPBPI Scraper in instances mode, followed by the collection of the five datapipes (as shown in Figure 3) is configured within the file:

DPIPE_HOME/scripts/BPIDP__collect_instances.pro

where *DPIPE_HOME* is the directory in which PI is installed.

Let's look at the five types of data that are retrieved when the HPBPI Scraper runs in instances mode, as shown in Figure 3 on page 37.

Flow Instances

When retrieving flow instances, the HPBPI Scraper accesses the following HPBPI database tables:

- `flows and flow_instance`

  These tables provides the list of completed flow instances for non-deleted flows.

  For each completed flow instance, the Scraper retrieves data such as the instance identifier, weight, start and complete times.

  The completed flow instances are selected by timestamp, using the column `EndTimeRecordedLongMillis`.

- Related data table

  For each completed flow instances, the HPBPI Scraper retrieves some of the related business data. By default, the Scraper sorts the related business data attributes alphabetically by name, and then retrieves the first 10 string-type attribute values.

  The type, and number, of business data attributes retrieved is configurable, refer to Business Data Handling on page 62 for more details.

The HPBPI Scraper writes this flow instance data to a CSF file. The CSF file is located in the directory:

DPIPE_HOME`/data/ImportData/Process_Insight`

where *DPIPE_HOME* is the directory in which PI is installed.

The name of the CSF file has the form:

`BPIDP_FLOWINST_`*timestamp*`.txt`

where *timestamp* is the time that the HPBPI Scraper created this CSF file, with the time expressed as the number of milliseconds since 1970.

The `BPIDP_FLOWINST` datapipe then runs `ee_collect` to process this CSF file and place the data into the PI database tables as shown in Figure 3 on page 37. Because it is possible for a flow to have more than one flow instance completing at the same time, the `R_BPI_FLOWINSTANCEDATA` table is defined as type `event`.

Node Instances

When retrieving node instances, the HPBPI Scraper accesses the following HPBPI database tables:

- `flows and nodes`

  These tables provides the list of nodes for non-deleted flows.

- `node_instance`

  For each completed node instance, the Scraper retrieves data such as the start and complete times.

  The completed node instances are selected by timestamp, using the column `EndTimeRecordedLongMillis`.

The HPBPI Scraper writes this node instance data to a CSF file. The CSF file is located in the directory:

DPIPE_HOME/data/ImportData/Process_Insight

where *DPIPE_HOME* is the directory in which PI is installed.

The name of the CSF file has the form:

BPIDP_NODEINST_*timestamp*.txt

where *timestamp* is the time that the HPBPI Scraper created this CSF file, with the time expressed as the number of milliseconds since 1970.

The BPIDP_NODEINST datapipe then runs ee_collect to process this CSF file and place the data into the PI database tables as shown in Figure 3 on page 37. Because it is possible for a node to have more than one node instance completing at the same time, the R_BPI_NODEINSTANCEDATA table is defined as type event.

Metric Values

When retrieving metric values, the HPBPI Scraper accesses the following HPBPI database tables:

- flows and metric_dim_metrics

  These tables provides the list of non-deleted metrics for non-deleted flows.

- metric_fact_values

  For each completed metric value, the Scraper retrieves data such as the start and complete times, the value of the metric and whether or not the metric deadline is overdue.

  The completed metric values are selected by timestamp, using the column LastUpdateTimeLongMillis.

The HPBPI Scraper writes this metric value data to a CSF file. The CSF file is located in the directory:

DPIPE_HOME/data/ImportData/Process_Insight

where *DPIPE_HOME* is the directory in which PI is installed.

The name of the CSF file has the form:

BPIDP_METRICVALUES_*timestamp*.txt

where *timestamp* is the time that the HPBPI Scraper created this CSF file, with the time expressed as the number of milliseconds since 1970.

The BPIDP_METRICVALUES datapipe then runs ee_collect to process this CSF file and place the data into the PI database tables as shown in Figure 3 on page 37. Because it is possible for a metric to have more than one metric value completing at the same time, the R_BPI_METRICVALUEDATA table is defined as type event.

Metric Statistics

When retrieving metric statistics, the HPBPI Scraper accesses the following HPBPI database tables:

- `flows and metric_dim_metrics`

  These tables provides the list of non-deleted metrics for non-deleted flows.

- `metric_fact_statistics`

  The Scraper retrieves a combination of both completed and active metric statistic information. For each recorded set of metric statistics, the Scraper retrieves data such as the count of completed metric instances, the average of all completed metric instances, the minimum and maximum of all completed metric instances, and the backlog and weight of all active metric instances.

  The completed metric statistics are selected by timestamp, using the column `TimeLongMillis`.

The HPBPI Scraper writes this metric statistics data to a CSF file. The CSF file is located in the directory:

DPIPE_HOME`/data/ImportData/Process_Insight`

where *DPIPE_HOME* is the directory in which PI is installed.

The name of the CSF file has the form:

`BPIDP_METRICSTATS_`*timestamp*`.txt`

where *timestamp* is the time that the HPBPI Scraper created this CSF file, with the time expressed as the number of milliseconds since 1970.

The `BPIDP_METRICSTATS` datapipe then runs `ee_collect` to process this CSF file and place the data into the PI database tables as shown in Figure 3 on page 37. Because it is possible for a metric to have more than one metric statistic occurring at the same time, the `R_BPI_METRICSTATSDATA` table is defined as type `event`.

Metric Alerts

When retrieving metric alerts, the HPBPI Scraper accesses the following HPBPI database tables:

- `flows`

  This table provides the list of non-deleted flows.

- `metric_fact_alerts`

  For each metric alert, the Scraper retrieves data such as the time the alert occurred, the level of the alert, and the flow instance that was associated with this alert (if any).

  The raised metric alerts are selected by timestamp, using the column `RaisedTimeLongMillis`.

The HPBPI Scraper writes this metric alert data to a CSF file. The CSF file is located in the directory:

DPIPE_HOME`/data/ImportData/Process_Insight`

where *DPIPE_HOME* is the directory in which PI is installed.

The name of the CSF file has the form:

```
BPIDP_METRICALERTS_timestamp.txt
```

where *timestamp* is the time that the HPBPI Scraper created this CSF file, with the time expressed as the number of milliseconds since 1970.

The BPIDP_METRICALERTS datapipe then runs ee_collect to process this CSF file and place the data into the PI database tables as shown in Figure 3 on page 37. Because it is possible for a metric to have more than one metric alert occurring at the same time, the R_BPI_METRICALERTDATA table is defined as type event.

History TimeStamp Logs

When the HPBPI Scraper runs in instances mode it retrieves all instance data that has occurred since the last time the Scraper ran. So how does the Scraper keep track of the last time it ran?

The Scraper maintains the last time stamp information in the following file:

```
DPIPE_HOME/lib/bpi_dpipe_history.properties
```

where *DPIPE_HOME* is the directory in which PI is installed.

This history file is created by the Scraper the first time it runs in instances mode.

The Scraper maintains the last time stamp separately for each of the five datapipe scrapes that it makes. These time stamps are then written to the history property file using the properties:

```
lastTimeStamp_FlowInst
lastTimeStamp_Nodeinst
lastTimeStamp_MetricStats
lastTimeStamp_MetricAlert
lastTimeStamp_MetricValue
```

The time stamps are saved as the number of milliseconds since 1970, however the Scraper also saves string versions of these time stamps in the history property file. These string versions of each time stamp are maintained purely for the troubleshooter to be able to see what time stamps are represented by the millisecond values. The string representations of the time stamps are maintained in the properties:

```
lastTimeStamp_FlowInst_StringDisplay
lastTimeStamp_Nodeinst_StringDisplay
lastTimeStamp_MetricStats_StringDisplay
lastTimeStamp_MetricValue_StringDisplay
lastTimeStamp_MetricAlert_StringDisplay
```

The Last Time Stamp

When the HPBPI Scraper runs in instances mode it retrieves all instance data that has occurred since the last time the Scraper ran. In actual fact, this is not quite true. As the Scraper completes each instance scrape, it keeps track of the timestamp of the last data record it processed. The Scraper then adds one millisecond to this timestamp value and saves this as the timestamp to use in any subsequent scrape.

So, for each of the five instance datapipe scrapes, the time stamp that is saved as the "lastTimeStamp" is one millisecond greater than the time of the last record processed.

For example, consider the scraping for flow instance data:

Suppose you run the instance Scraper every hour.

At 5pm, the Scraper retrieves 15 flow instances, where the latest of these instances is time stamped as completing at 4:15pm.

The `lastTimeStamp_FlowInst` property in the history property is set to one millisecond past 4:15pm.

At 6pm the Scraper looks for all flow instances that have completed since one millisecond past 4:15, up until 6pm.

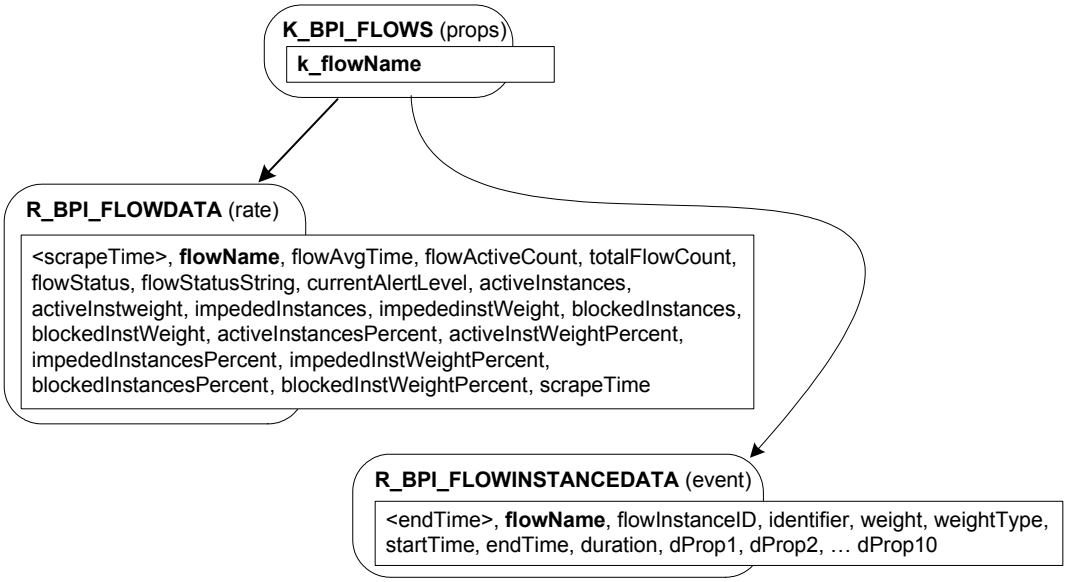
PI Base Table Structures

Here are the details of the PI database tables populated by the HPBPI Report Pack datapipes.

Flow Tables

Figure 4 shows the data tables populated by the datapipes `BPIDP_FLOWS` and `BPIDP_FLOWINST`.

Figure 4 Flow Tables



Let's now list each table, describing each data column.

K_BPI_FLOWS

The flows property table.

Table 2 K_BPI_FLOWS

Column Name	Data Type	Length	Description
k_flowName	char_string	120	Name of the flow.

R_BPI_FLOWDATA

The rate table that holds the flow status data.

Table 3 R_BPI_FLOWDATA

Column Name	Data Type	Length	Description
scrapeTime	unix_time		Time of this scrape of the HPBPI flows table. This value is also passed as the ta_period.
flowName	char_string	120	Name of the flow.
flowAvgTime	float		Average time of all completed flow instances for this flow.
flowActiveCount	integer		Number of flow instances active at the time of this scrape.
totalFlowCount	integer		Total number of flow instances – includes completed and currently active instances.
flowStatus	integer		Business status of the flow – as a percentage, where: 100 = Active 50 = Impeded 0 = Blocked
flowStatusString	char_string	12	Business status of the flow. Values are: Active, Impeded and Blocked.
currentAlertLevel	integer		Maximum alert from all flow metrics, where: 1 = Normal 2 = Warning 3 = Minor 4 = Major 5 = Critical
activeInstances	integer		Number of currently active flow instances.
activeInstWeight	float		Total weight of currently active flow instances.
impededInstances	integer		Number of currently impeded flow instances.
impededInstWeight	float		Total weight of currently impeded flow instances.
blockedInstances	integer		Number of currently blocked flow instances.
blockedInstWeight	float		Total weight of currently blocked flow instances.

Table 3 R_BPI_FLOWDATA (cont'd)

Column Name	Data Type	Length	Description
activeInstancesPercent	float		Percentage currently active flow instances.
activeInstWeightPercent	float		Percentage weight of currently active flow instances.
impededInstancesPercent	float		Percentage currently impeded flow instances.
impededInstWeightPercent	float		Percentage weight of currently impeded flow instances.
blockedInstancesPercent	float		Percentage currently blocked flow instances.
blockedInstWeightPercent	float		Percentage weight of currently blocked flow instances.

R_BPI_FLOWINSTANCEDATA

The event table that holds the completed flow instance data.

Table 4 R_BPI_FLOWINSTANCEDATA

Column Name	Data Type	Length	Description
flowName	char_string	120	Name of the flow.
flowInstanceID	char_string	36	The HPBPI internal ID for the flow instance.
identifier	char_string	120	Identifier for the flow instance.
weight	float		Weight of the flow instance.
weightType	char_string	120	The type of the weight.
startTime	unix_time		Start time of the flow instance.
endTime	unix_time		End time of the flow instance. This value is also passed as the ta_period.
duration	float		Difference between the start and end time.
dProp1	char_string	50	Business data property 1.
dProp2	char_string	50	Business data property 2
dProp3	char_string	50	Business data property 3
dProp4	char_string	50	Business data property 4
dProp5	char_string	50	Business data property 5
dProp6	char_string	50	Business data property 6
dProp7	char_string	50	Business data property 7

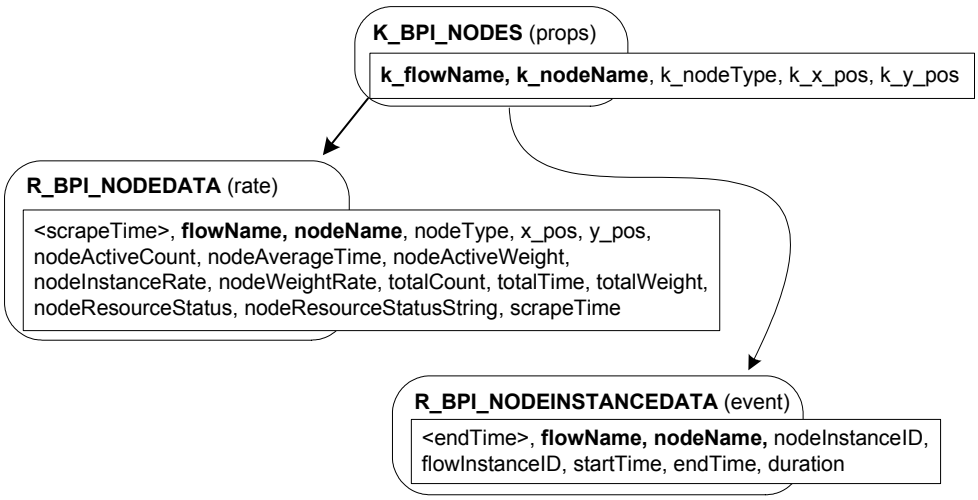
Table 4 R_BPI_FLOWINSTANCEDATA (cont'd)

Column Name	Data Type	Length	Description
dProp8	char_string	50	Business data property 8
dProp9	char_string	50	Business data property 9.
dProp10	char_string	50	Business data property 10.

Node Tables

Figure 5 shows the data tables populated by the datapipes `BPIDP_NODES` and `BPIDP_NODEINST`.

Figure 5 Node Tables



Let's now list each table, describing each data column.

K_BPI_NODES

The nodes property table.

Table 5 K_BPI_NODES

Column Name	Data Type	Length	Description
k_flowName	char_string	120	Name of the flow.
k_nodeName	char_string	120	Name of the node.
k_nodeType	char_string	12	Type of the node. Values are Start, Work, End, Junction.
k_x_pos	integer		X coordinate of the node within the HPBPI flow diagram.
k_y_pos	integer		Y coordinate of the node within the HPBPI flow diagram.

R_BPI_NODEDATA

The rate table that holds the node status data.

Table 6 R_BPI_NODEDATA

Column Name	Data Type	Length	Description
scrapeTime	unix_time		Time of each scrape of the HPBPI Nodes table. This value is also passed as the ta_period.
flowName	char_string	120	Name of the flow.
nodeName	char_string	120	Name of the node.
nodeType	char_string	12	Type of the node. Values are Start, Work, End and Junction.
x_pos	integer		X co-ordinate of the node within the HPBPI flow diagram.
y_pos	integer		Y co-ordinate of the node within the HPBPI flow diagram.
nodeActiveCount	integer		Number of active node instances in this node.
nodeAverageTime	float		Average time of all completed node instances.
nodeActiveWeight	float		Total value of the weight of all flow instances currently in this node.
nodeInstanceRate	float		Measure of the number of instances per hour that are currently being processed at this node.
nodeWeightRate	float		Measure of the weight per hour that is currently being processed at this node.
totalCount	float		Total number of times this node has been started.
totalTime	float		Accumulated active time of all instances of this node.

Table 6 R_BPI_NODEDATA (cont'd)

Column Name	Data Type	Length	Description
totalWeight	float		Accumulated weight of all instances of this node.
nodeResourceStatus	integer		Status of any underlying IT, where: 100 = Normal 80 = Warning 60 = Minor 40 = Major 20 = Critical 0 = Unset
nodeResourceStatusString	char_string	12	Status of any underlying IT. Value are Normal, Warning, Minor, Major, Critical and Unset.

R_BPI_NODEINSTANCEDATA

The event table that holds the completed node instance data.

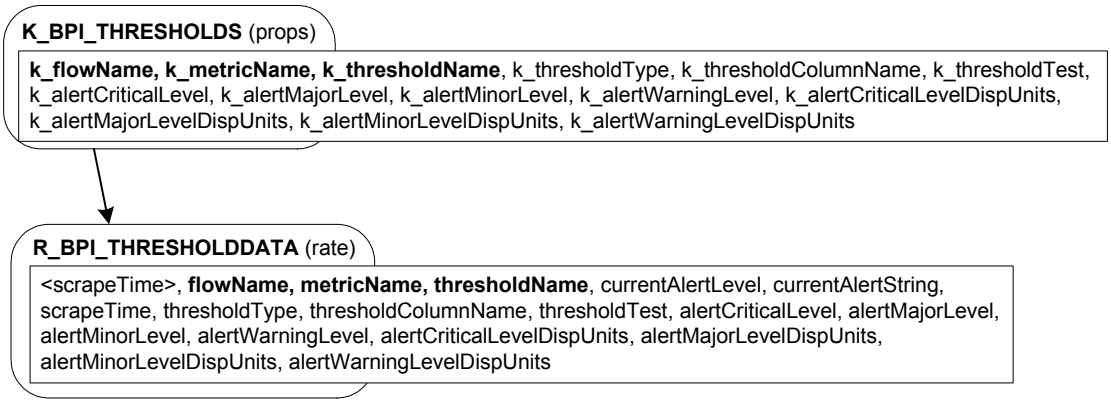
Table 7 R_BPI_NODEINSTANCEDATA

Column Name	Data Type	Length	Description
flowName	char_string	120	Name of the flow.
nodeName	char_string	120	Name of the node.
nodeInstanceID	char_string	36	The HPBPI internal ID for the node instance.
flowInstanceID	char_string	36	The HPBPI internal ID for the flow instance.
startTime	unix_time		Start time of the node instance.
endTime	unix_time		End time of the node instance. This value is also passed as the ta_period.
duration	float		Difference between the start and end times.

Threshold Tables

Figure 6 shows the data tables populated by the datapipe `BPIDP_THRESHOLDS`.

Figure 6 Threshold Tables



Let's now list each table, describing each data column.

K_BPI_THRESHOLDS

The thresholds property table.

Table 8 K_BPI_THRESHOLDS

Column Name	Data Type	Length	Description
k_flowName	char_string	120	Name of the flow.
k_metricName	char_string	120	Name of the metric.
k_thresholdName	char_string	120	Name of the threshold
k_thresholdType	char_string	12	Type of threshold. Values are instance, active, completed and total.
k_thresholdColumnName	char_string	40	The column that the threshold applies to.
k_thresholdTest	char_string	12	Type of test applied to the threshold.
k_alertCriticalLevel	float		Value of the critical threshold – in the same units as the metric.
k_alertMajorLevel	float		Value of the major threshold – in the same units as the metric.
k_alertMinorLevel	float		Value of the minor threshold – in the same units as the metric.
k_alertWarningLevel	float		Value of the warning threshold – in the same units as the metric.
k_alertCriticalLevelDispUnits	char_string	12	Display units of the critical threshold – as used within the metric definer.

Table 8 K_BPI_THRESHOLDS (cont'd)

Column Name	Data Type	Length	Description
k_alertMajorLevelDispUnits	char_string	12	Display units of the major threshold – as used within the metric definer.
k_alertMinorLevelDispUnits	char_string	12	Display units of the minor threshold – as used within the metric definer.
k_alertWarningLevelDispUnits	char_string	12	Display units of the warning threshold – as used within the metric definer.

R_BPI_THRESHOLDDATA

The rate table that holds the threshold status data.

Table 9 R_BPI_THRESHOLDDATA

Column Name	Data Type	Length	Description
scrapeTime	unix_time		Time of each scrape of the threshold table. This value is also passed as the ta_period.
flowName	char_string	120	Name of the flow.
metricName	char_string	120	Name of the metric.
thresholdName	char_string	120	Name of the threshold
currentAlertLevel	integer		Current alert level, where: 0 = Unset 1 = Normal 2 = Warning 3 = Minor 4 = Major 5 = Critical
currentAlertString	char_string	12	Current alert level. Values are Unset, Normal, Warning, Minor, Major and Critical.
thresholdType	char_string	12	Type of threshold. Values are instance, active, completed and total.
thresholdColumnName	char_string	40	The column that the threshold applies to.
thresholdTest	char_string	12	Type of test applied to the threshold.
alertCriticalLevel	float		Value of the critical threshold – in the same units as the metric.

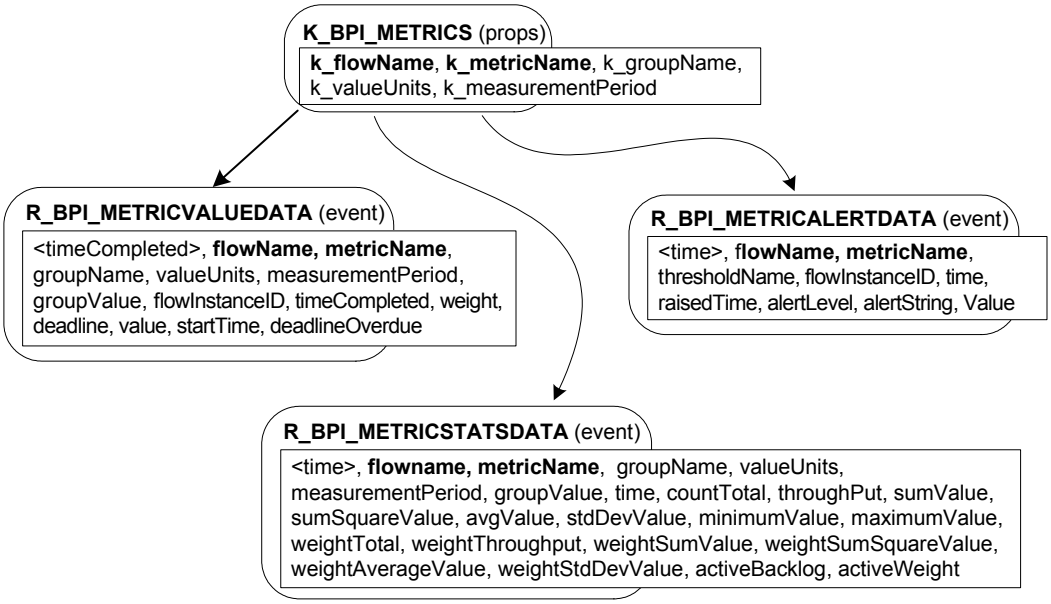
Table 9 R_BPI_THRESHOLDDATA (cont'd)

Column Name	Data Type	Length	Description
alertMajorLevel	float		Value of the major threshold – in the same units as the metric.
alertMinorLevel	float		Value of the minor threshold – in the same units as the metric.
alertWarningLevel	float		Value of the warning threshold – in the same units as the metric.
alertCriticalLevelDispUnits	char_string	12	Display units of the critical threshold – as used within the metric definer.
alertMajorLevelDispUnits	char_string	12	Display units of the major threshold – as used within the metric definer.
alertMinorLevelDispUnits	char_string	12	Display units of the minor threshold – as used within the metric definer.
alertWarningLevelDispUnits	char_string	12	Display units of the warning threshold – as used within the metric definer.

Metric Tables

Figure 7 shows the data tables populated by the datapipes `BPIDP_METRICVALUES`, `BPIDP_METRICSTATS` and `BPIDP_METRICALERTS`.

Figure 7 Metric Tables



Let's now list each table, describing each data column.

K_BPI_METRICS

The metric property table.

Table 10 K_BPI_METRICS

Column Name	Data Type	Length	Description
k_flowName	char_string	120	Name of the flow.
k_metricName	char_string	120	Name of the metric defined for this flow.
k_groupName	char_string	120	Name of the group defined for this metric.
k_valueUnits	char_string	12	All metric values, statistics and thresholds are in this unit of measure.
k_measurementPeriod	integer		Collection interval for the metric statistics – in seconds.

R_BPI_METRICVALUEDATA

The event table that holds the completed metric value data.

Table 11 R_BPI_METRICVALUEDATA

Column Name	Data Type	Length	Description
flowName	char_string	120	Name of the flow.
metricName	char_string	120	Name of the metric defined for this flow.
groupName	char_string	120	Name of the group defined for this metric.
valueUnits	char_string	12	All metric values, statistics and thresholds are in this unit of measure.
measurementPeriod	integer		Collection interval for the metric statistics – in seconds.
groupValue	char_string	180	Value of the group for which this metric value belongs.
flowInstanceID	char_string	36	The HPBPI internal ID for the flow instance that is associated with this metric value.
timeCompleted	unix_time		Time the metric instance, that produced this metric value, completed. This value is also passed as the ta_period.
weight	float		The weight of the metric.

Table 11 R_BPI_METRICVALUEDATA (cont'd)

Column Name	Data Type	Length	Description
deadline	unix_time		If this metric value is for a deadline metric, this is the deadline date.
value	float		The value of the metric.
startTime	unix_time		Time the metric instance, that produced this metric value, started.
deadlineOverdue	float		If this metric value is for a deadline metric, this is the number of seconds outside the deadline. Positive means overdue. Negative means ahead of schedule.

R_BPI_METRICSTATSDATA

The event table that holds the completed metric statistics data.

Table 12 R_BPI_METRICSTATSDATA

Column Name	Data Type	Length	Description
flowName	char_string	120	Name of the flow.
metricName	char_string	120	Name of the metric defined for this flow.
groupName	char_string	120	Name of the group defined for this metric.
valueUnits	char_string	12	All metric values, statistics and thresholds are in this unit of measure.
measurementPeriod	integer		Collection interval for the metric statistics – in seconds.
groupValue	char_string	180	Value of the group for which this metric value belongs.
time	unix_time		The time of this metric statistic. This value is also passed as the ta_period.
countTotal	integer		Total number of completed metric instances used to calculate this metric statistic interval.
throughput	float		The number of metric instances per hour based on this sample.
sumValue	float		Total of the values of the completed metric instance in this statistic interval.

Table 12 R_BPI_METRICSTATSDATA (cont'd)

Column Name	Data Type	Length	Description
sumSquareValue	float		Total of the squares of the completed metric values in this statistic interval.
avgValue	float		Average of the completed metric instance values in this statistic interval.
stdDevValue	float		Standard deviation of the completed metric instance values in this statistic interval
minimumValue	float		Minimum completed metric instance value in this statistic interval.
maximumValue	float		Maximum completed metric instance value in this statistic interval.
weightTotal	float		Total of the weights of the completed metric instances in this statistic interval.
weightThroughput	float		The weight per hour of the completed metric instances based on this sample.
weightSumValue	float		Total of all completed metric weights in this statistic interval.
weightSumSquareValue	float		Total of the squares of the completed metric weights in this statistic interval.
weightAverageValue	float		Average of the completed metric instance weights in this statistic interval.
weightStdDevValue	float		Standard deviation of the completed metric instance weights in this statistic interval.
activeBacklog	integer		Total number of active metric instances in this statistic interval.
activeWeight	float		Total of the weights of the active metric instances in this statistic interval.

R_BPI_METRICALERTDATA

The event table that holds the metric alert data.

Table 13 R_BPI_METRICALERTDATA

Column Name	Data Type	Length	Description
flowName	char_string	120	Name of the flow.
metricName	char_string	120	Name of the metric defined for this flow.
thresholdName	char_string	120	Name of the threshold defined for this metric.
flowInstanceID	char_string	36	The HPBPI internal ID for the flow instance that caused with this alert.
time	unix_time		Time the alert actually occurred in the real world. This value is also passed as the ta_period.
raisedTime	unix_time		Time HPBPI raised this alert.
alertLevel	integer		The level of alert, where: 1 = Normal 2 = Warning 3 = Minor 4 = Major 5 = Critical
alertString	char_string	12	The level of alert. Values are Normal, Warning, Minor, Major and Critical.
value	float		The value (of the metric) that caused this alert.

Configuring the HPBPI Scraper

As described in Chapter 2, Installation, the HPBPI Scraper needs minimal configuration. Indeed, the only configuration details required are the details of the connection to the HPBPI database.

The HPBPI Scraper reads its configuration details from the following file:

DPIPE_HOME/lib/bpi_dpipe.properties

where *DPIPE_HOME* is the directory in which PI is installed.

When you make changes to the bpi_dpipe.properties file, this affects the next invocation of the HPBPI Scraper.

Let's now consider the list of configurable properties for the HPBPI Scraper.

HPBPI Database Connection

The properties that set the HPBPI database connection details are as follows:

- dbDriver
- dbProtocol
- dbUser
- dbPasswordEncoding
- dbPassword

These are all described in Chapter 2, Installation, in the section HPBPI Database Connection on page 19.

Logging

By default, the HPBPI Scraper writes all log output to log files. Whether the HPBPI Scraper logs to a file or stdout is configurable. You can also configure the log level to either increase or decrease the detail of the log messages.

The logging properties are as follows:

- logOutputFile
- logOutputFileSuffix
- logOutputLevel
- logTimeStamps

Log Level

The logOutputLevel property allows you to set the level at which the HPBPI Scraper outputs log messages:

There are three available log levels:

- ERROR

  Only output ERROR log messages.

- `INFO`

  Output log messages that are of level `INFO` or `ERROR`.

  `INFO` log messages allow you to see the values that the HPBPI Scraper has read from the configuration properties file.

- `DEBUG`

  Output log messages that are of level `DEBUG`, `INFO` or `ERROR`.

  This log level allows you to see the actual SQL calls issued by the HPBPI Scraper.

For example:

```
logOutputLevel = DEBUG
```

configures the Scraper to run in `DEBUG` mode and output `DEBUG` log messages.

Specifying the Log Output Location

The following two properties allow you to specify the location of any log output:

- `logOutputFile`
- `logOutputFileSuffix`

If the property `logOutputFile` is commented-out then all log messages are written to the HPBPI Scraper's standard output/error devices. `ERROR` log messages are written to stderr, and all other log messages are written to stdout.

If the property `logOutputFile` is set, then this value is used to form a file name to hold the log output from the HPBPI Scraper. The actual name of the log file is built by concatenating three things:

1. The value of the `logOutputFile` property.
2. The mode that the HPBPI Scraper is running in - `definitions` or `instances`.
3. The value of the `logOutputFileSuffix` property.

For example, suppose you have the following property settings:

```
logOutputFile = c:\log\myFile_
logOutputFileSuffix = .abc
```

When you run the HPBPI Scraper in `definitions` mode, the log message are written to the file:

```
c:\log\myFile_definitions.abc
```

If the property `logOutputFileSuffix` is not set then it defaults to `.log`.

If the value of the `logOutputFile` property starts with the token `{DPIPE_HOME}` then this token is replaced with the value found in the `DPIPE_HOME` environment variable

Log Message Time Stamps

By default, all log message contain a time stamp to show when they were produced. If you wish to disable the inclusion of these time stamps then you can set the `logTimeStamps` property to `false`.

The `logTimeStamps` property defaults to `true`.

Test Configuration Mode

If you want to test your HPBPI Scraper property settings, including the database connection, without the HPBPI Scraper actually accessing any data then you can set the property:

```
testConfigAndExit
```

This property is normally commented out and defaults to `false`.

To test your HPBPI property file and the database connection, you can do the following steps:

1. Un-comment the `testConfigAndExit` property and set it to `true`:

   ```
   testConfigAndExit = true
   ```

2. Run the HPBPI Scraper standalone.
   - Start up a command window

     For UNIX, log in as the user `trendadm`
   - The run the HPBPI Scraper as follows:

     DPIPE_HOME`/bin/bpi_dpipe -mode definitions`

     where *DPIPE_HOME* is the directory in which PI is installed.

     The actual mode you choose (`definitions` or `instances`) does not matter as no HPBPI data will be accessed.
3. Check the log output.

   You can see if any errors occurred.

   If you are running with log level `INFO` then you can also see the property values as read from the properties file, and you are be able to see whether the database connection was successful.

When you are finished testing your HPBPI Scraper configuration, make sure that you comment-out the `testConfigAndExit` property, or set it to `false`.

Epoch Start Date

By default, when the HPBPI Scraper runs in `instances` mode for the first time, it uses the current time as the “last time”. In other words, the Scraper asks for all completed instance data since now. This means that, by default, the first run of the Scraper in `instances` mode returns no data. Subsequent runs of the Scraper retrieve the instance data as it occurs. However, you can specify a starting date where you tell the Scraper to select from HPBPI data that is on or after a specified date/time.

Specifying a start date/time only affects the Scraper when it is run in `instances` mode, and it only affects the first time the Scraper runs in this mode. All subsequent executions of the Scraper in `instances` mode use the date/time values that were saved by the previous execution. Refer to History TimeStamp Logs on page 41 for more details.

To specify a starting date/time for the first execution of the Scraper in `instances` mode, you can specify the following properties:

- `epochStartTimestampFormat`

  This property allows you to specify a `java.text.SimpleDateFormat` compatible date format string to describe the value you set in the `epochStartTimestamp` property.

  The default setting for the `epochStartTimestampFormat` property is `yyyy-MM-dd HH:mm:ss.SSS`

  The time zone defaults to UTC.

- `epochStartTimestamp`

  This lets you specify a date/time which specifies the start time for the first scrape of HPBPI instances.

  By default, the `epochStartTimestamp` property is set to the time you first run the HPBPI Scraper in `instances` mode.

Example Configurations

If you want to scrape HPBPI instance data from on or after 23 June 2007 at 2:30pm UTC, set the `epochStartTimestamp` property as follows:

```
epochStartTimestamp = 2007-06-23 14:30:00.000
```

If you want to scrape HPBPI instance data from on or after 23 November 2006 at 12:04:33.998 in the time zone of UTC-7, you need to set an epoch time stamp format that includes a time zone and then specify this time zone with the epoch start time, as follows:

```
epochStartTimestampFormat = yyyy-MM-dd HH:mm:ss.SSS Z
epochStartTimestamp = 2006-11-23 12:04:33.998 -0700
```

If you want to test whether you have specified your epoch time stamp and format correctly, then you can use the `testConfigAndExit` property with the logging level set to `INFO`.

Large Select Statements

When the HPBPI Scraper runs in `instances` mode it issues select statements that can potentially retrieve very large amounts of data from the HPBPI database.

There is a slight possibility that the data retrieved from HPBPI in any single select statement might be so large that the HPBPI Scraper might be unable to process it. Alternatively, the select statement might take so long to process that it causes some performance issues within the HPBPI database.

If you experience any such problems then you have the option to configure the Scraper (when running in `instances` mode) to split all select statements into smaller blocks of time.

For example, suppose you are running the Scraper in `instances` mode for the very first time and you have specified an epoch start date/time of January 1st 1970. In this situation the Scraper is going to ask for all completed node instances from January 1st 1970 up until now, and ask for this in a single SQL select statement. You can configure the Scraper to split this single select statement into a series of smaller sub-select statements.

The properties that allow you to split the single SQL selects are as follows:

- `numberOfInstanceSelectBlocks`

  The Scraper will break each single select statement into this many sub-selects.

  This property defaults to the value `1`.

- `selectSleepTime`

  You specify the time, in milliseconds, that you would like the Scraper to sleep in between issuing each sub-select statement.

  The default sleep time is `0`.

Let's continue the example where you are running the Scraper in `instances` mode for the very first time, specifying an epoch start time of January 1st 1970, and the Scraper is retrieving all completed node instances. Suppose the current time is July 24th 2007 13:30:07.704 UTC. If you were to run the Scraper it would scrape for all node instances by issuing a single select statement looking for all completed node instances between the times January 1st 1970 00:00:00 until July 24th 2007 13:30:07.704.

Suppose you decide to split this single select statement into smaller blocks by setting the properties as follows:

```
numberOfInstanceSelectBlocks = 5
selectSleepTime = 1000
```

Now, when the Scraper is scraping for completed node instances it divides the overall scrape time into five equal blocks and then loops through issuing each select statement covering each of these time spans. If the overall time span doesn't divide exactly by five then there will be an additional select statement to cover the remaining time span. For example, the Scraper would issue select statements with the following series of time periods:

- 01 Jan 1970 00:00:00.000 <= completed time < 06 Jul 1977 17:06:01.540
- 06 Jul 1977 17:06:01.540 <= completed time < 09 Jan 1985 10:12:03.080
- 09 Jan 1985 10:12:03.080 <= completed time < 15 Jul 1992 03:18:04.620
- 15 Jul 1992 03:18:04.620 <= completed time < 18 Jan 2000 20:24:06.160
- 18 Jan 2000 20:24:06.160 <= completed time < 24 Jul 2007 13:30:07.700
- 24 Jul 2007 13:30:07.700 <= completed time < 24 Jul 2007 13:30:07.704

with a sleep for 1000 milliseconds (1 second) between each select statement.

If you set the `numberOfInstanceSelectBlocks` and `selectSleepTime` properties these affect all selects for the HPBPI Scraper when run in `instances` mode. This means that the retrieval of completed flow instances, completed node instances, metric values, metric statistics and metric alerts are each split into `numberOfInstanceSelectBlocks` select statements.

Null Values in Strings

The PI utility `trend_sum` can only trend data that does not contain null values. For most of the HPBPI data brought across to PI this is not a problem, because attributes such as flow name, and metric name are not allowed to be null within HPBPI. However, there are some attributes within HPBPI that can contain null values, and if you want to allow PI users to set up trend sums on these attributes then you need to provide some sort of substitution for these null values.

The HPBPI Scraper automatically substitutes the following HPBPI attributes if they have a null value:

- Metric group name
- Metric group value
- Flow instance identifier
- Business data property

Let look at these in more detail...

Metric Group Name

If the HPBPI Scraper sees a metric group name with a null value it sets that metric group name to the string `_No-Groups_`.

You can alter this default behavior by setting the property `metricGroupNameNullDefault`.

You should not need to modify this property.

If you do modify this property then you need to edit the PI reports that display metric group names, as these reports are coded to test for the string `_No-Groups_` and display an empty string within the report.

Metric Group Value

Within HPBPI, a metric group value may have a null value. This can mean one of two things:

1. A blank metric group value.

   If a metric is grouped then it is possible that one of the values for that group is null. The HPBPI Scraper refers to this as a blank metric group value.

   If the HPBPI Scraper sees a blank metric group value, the null value is set to the string `_Blank_`.

   You can alter this default behavior by setting the property `metricGroupValueBlankDefault`.

   You should not need to modify this property.

2. The overall value for all metric groups.

   If a metric is grouped then HPBPI maintains separate metric data for each of these groups as well as overall metric data for all these groups combined. This combined metric data records are written with a group value of null.

   Also, if a metric is not grouped, then the metric data within the HPBPI database is written with the group value set to null.

   If the HPBPI Scraper sees a null metric group value that is for all metric groups, the null value is set to the string `_All_`.

   You can alter this default behavior by setting the property `metricGroupValueNullDefault`.

   You should not need to modify this property.

Flow Identifier

It is possible for the identifier of an HPBPI flow instance to be a null value. If the HPBPI Scraper sees a flow identifier with a null value it sets that identifier to the string `_No-Identifier_`.

You can alter this default behavior by setting the property `flowInstIdentifierNullDefault`.

You should not need to modify this property.

Business Data Attributes

When the HPBPI Scraper is retrieving flow instances, for each flow instance, the Scraper also retrieves up to 10 of the related business data attributes and brings these across as string values. It is possible that some of these business data attributes have a null value.

If a business data attribute has a null value then the HPBPI Scraper defaults the value to six dash (-) characters. In other words, a null is replaced by the string `------`.

The property `flowInstBusDataAttrBlankFillerChar` defines the character to be used as the filler character. This property defaults to the dash (-) character.

The property `busDataAttrDefFillLength` defines the number of these filler characters that are used. (Refer to Business Data Filler on page 63.)

Business Data Handling

When the HPBPI Scraper is retrieving flow instances, for each flow instance, the Scraper also retrieves up to 10 of the string business data attributes. This behavior is configurable.

Type of Business Data

By default, the Scraper only brings across business data attributes that are defined within HPBPI as type `String`. The `busDataAttrType` property lets you configure the Scraper to bring across more than just string business data attributes.

The `busDataAttrType` property can be set to the following values:

- `String`

  The Scraper only brings across business data attributes that are defined within HPBPI as type `String`.

  This is the default setting.

- `String-Date`

  The Scraper brings across business data attributes that are of type `String` or `Date`.

- `String-Date-Integer`

  The Scraper brings across business data attributes that are of type `String`, `Date` or `Integer`.

- `All`

  The Scraper brings across business data attributes that are of any data type.

When the business data attribute values are stored within the PI data tables, all values are stored as character strings.

Amount of Business Data

By default the HPBPI Scraper brings across up to 10 business data attributes.

The Scraper first sorts the list of defined business data attributes alphabetically by name. The Scraper then loops through and selects the first 10 attributes that match the data type(s) specified by the `busDataAttrType` property.

For each selected business data attribute, the Scraper retrieves up to the first 50 characters.

The number of business data attributes retrieved can be configured by the property `busDataAttrCount`. This property defaults to the value `10`.

The maximum number of characters retrieved for each business data attribute is configured by the property `busDataAttrMaxLength`. This property defaults to the value `50`.

You should not need to modify these properties.

If you do modify these properties then you must also modify all PI table definitions that include these business data attributes. This would include the `R_BPI_FLOWINSTANCEDATA` table defined in the file `BPIDP_tables_FLOWINST.teel` and any associated views.

Business Data Filler

When a business data attribute value is null, the HPBPI Scraper defaults the value to six dash (`-`) characters. In other words, a null is replaced by the string `------`.

The property `flowInstBusDataAttrBlankFillerChar` defines the character to be used as the filler character. (Refer to Business Data Attributes on page 62.)

The property `busDataAttrDefFillLength` defines the number of filler characters that are used.

The `busDataAttrDefFillLength` property defaults to the value `6`.

You can modify the number of filler characters by setting the `busDataAttrDefFillLength` property.

JDBC Type Definitions

When the HPBPI Scraper processes business data attributes it maps the HPBPI data types to JDBC data types. These mappings are all set up by default so you should never need to alter them.

The properties that allow you to alter the HPBPI data type to JDBC data type mappings are as follows:

- `busDataAttrStringJavaType`

  This property lists the JDBC data types that map to the HPBPI data type `String`. The default values are `-9` and `12`.

- `busDataAttrDateJavaType`

  This property lists the JDBC data types that map to the HPBPI data type `Date`. The default values are `91, 92` and `93`.

- `busDataAttrIntegerJavaType`

  This property lists the JDBC data types that map to the HPBPI data type `Integer`. The default values are `2` and `4`.

- `busDataAttrCurrencyJavaType`

This property lists the JDBC data types that map to the HPBPI data type `Currency`. The default values are `2` and `3`.

The HPBPI Scraper uses these mappings when it processes the business data attributes for a flow instance. For example, if the `busDataAttrType` property is set to `String-Date`, then the `busDataAttrStringJavaType` and `busDataAttrDateJavaType` properties enables the Scraper to know which JDBC data types satisfy these `String` and `Date` attribute types.

You should not need to alter these property settings.

Metric Group Value Truncation

Within HPBPI, a metric's group value attribute is defined as being a string up to 256 characters in length.

Within the PI database the group value attribute is combined with the flow name, metric name and group name attributes to produce a combined key when producing summary tables for metric statistics and metric values. Unfortunately, there is a limit within the Sybase database management system that limits the overall size of a combined key. So, the HPBPI Scraper is written to truncate the metric group value attribute at 180 characters in length.

This means that if your HPBPI database does contain metric groups where their group values do not differ within the first 180 characters, then these group values will come across to PI and be reported as if they were within the same metric group value.

The property `metricGroupValueMaxLength` allows you to alter the length of the group value attribute, however, if you change this then you also need to change the definition of the group value attribute in all PI tables that define this attribute. This would include the `R_BPI_METRICSTATSDATA` and `R_BPI_METRICVALUEDATA` tables and associated views.

The `metricGroupValueMaxLength` property defaults to the value `180`.

JDBC ResultSet Fetch Size

When the HPBPI Scraper is retrieving instance data, the number of data records returned for a single select statement can be very large. To ensure that the Java Virtual Machine (JVM) running the Scraper is not overloaded with all the selected records at once, the Scraper tells the JDBC subsystem to bring the physical data across in batches of up to 20000 records. The Scraper still issues a single select statement when it asks to retrieve a set of data, however the underlying JDBC subsystem pages the physical transmission of data records.

The `jdbcFetchSize` property enables you to alter the number of physical records brought across to the JVM at any one time.

The `jdbcFetchSize` property defaults to the value `20000`.

Setting the `jdbcFetchSize` property to zero `(0)` tells the JDBC subsystem to bring all records across at once.

Metric Statistic Duplicate Checking

Within HPBPI, if a user makes a change to a metric definition (from within the HPBPI Metric Definer) and chooses to "update" the metric, then the HPBPI Metric Engine starts to collect duplicate metric statistics. This is all cleared up if/when the HPBPI user chooses to "replace" the metric. Unfortunately, if the HPBPI Scraper was to bring across these duplicate metric statistic entries this would cause problems when the PI reports try to graph this data.

The HPBPI Scraper has been written to filter out any duplicate metric statistic entries.

However, to filter out these duplicate entries requires the Scraper to maintain an in-memory list of metric statistic IDs and it is possible that this list could grow so large that it causes the Java Virtual Machine (JVM) to throw an exception. If you experience this problem you can choose to disable the filtering of duplicate metric statistics.

To disable the filtering of duplicate metric statistics, set the `duplicateRecordCheck` property to `false`.

Scrape Time Lag

When the HPBPI Scraper runs in `instances` mode, it select all instance data that has occurred since the last time it scraped each HPBPI data table. The Scraper uses an attribute within each instance-based data table to do this selection.

Let's consider the example where the Scraper is selecting completed flow instances from the `flow_instance` table:

When a flow instance completes, the `status` attribute for the flow instance is set to `Completed` and the `EndTimeRecordedLongMillis` attribute is set to the time (as milliseconds since 1970) that this flow instance completed.

Now, when the Scraper comes to scrape the `flow_instance` table, the Scraper issues a select statement of the form:

```
select <instance data> from flow_instance
  where status = 'Completed'
    and EndTimeRecordedLongMillis >= <the last scrape time>
    and EndTimeRecordedLongMillis  < <now time>
```

That is, the select statement is looking for all flow instances that completed since the last time the Scraper scraped this table, up until now.

However, on a busy HPBPI server it is possible that there is a slight delay between the HPBPI Impact Engine assigning an `EndTimeRecordedLongMillis` value and this flow instance record then being committed to the HPBPI database. This means that at the time the Scraper issues its select statement, there could be a flow instance record, with an `EndTimeRecordedLongMillis` value less than `<now time>`, that has not yet been committed to the database. In other words, it could be possible that the Scraper misses this record.

To allow for the possibility that there could be a slight delay between the HPBPI Server assigning a timestamp to an instance record and that record then being committed to the HPBPI database, the HPBPI Scraper makes use of a "time lag". When the Scraper runs in `instances` mode, it first retrieves the current time from the HPBPI database. The Scraper then subtracts one minute from this time, and then uses this resultant time as the "now" time.

The amount of "lag time" subtracted from the actual "now" time is configured by the property `scrapeTimeLag`.

The scrapeTimeLag property specifies the time lag in milliseconds, and defaults to the value 60000 (which is 1 minute).

Flow Instance Active/AtRisk/Blocked Counts

When the HPBPI Scraper runs in definitions mode, it retrieves the status information for each deployed flow, node and threshold.

When retrieving the status information for each deployed flow, the Scraper collects three basic things:

1. The overall flow status
2. The current highest metric alert for the flow
3. The number of flow instances that are in the states Blocked, At Risk and Healthy

The first two steps do not involve the retrieval of many data records. However, step 3 can possibly be an issue for HPBPI systems with many thousands of flow instances.

For an HPBPI system with many thousands of flow instances, step 3 could take some time and affect the performance of your HPBPI database. For this reason, the Scraper can be configured not to carry out step 3.

To configure the Scraper not to carry out step 3 and to therefore not determine the number of flow instances that are in the states Blocked, At Risk and Healthy, you set the property flowsCalcStateCounts to false.

HPBPI Database Date/Time Stored Procedure

When the HPBPI Scraper runs, it calls an HPBPI stored procedure within the HPBPI database to determine the current database time. This time is then used when scraping instance data from the HPBPI database.

The HPBPI stored procedure that is called to determine the database time is called METRIC_GET_DATABASE_TIME_MSEC, and this stored procedure is installed when you install HPBPI.

If the name of this stored procedure was ever changed then the property bpiDBTimeSP can be set to call the new procedure name.

You should not need to alter this property.

Date Format in CSF Files

The HPBPI Scraper uses the property defautDateOutputFormat to specify the format to use when writing out date/time fields to the CSF files.

You should not need to modify this property. If you do modify this property then you must also modify the TaPeriodFormat property in all the BPIDP_collect_*.teel files.

The defautDateOutputFormat property default to the format yyyyMMdd HH:mm:ss

This default format matches the PI TaPeriodFormat format type 7 and this is the type specified in the BPIDP_collect_*.teel files.

CSF Output Directory

The property `outputDir` allows you to specify the directory in which the HPBPI Scraper writes the CSF files.

If the value of the `outputDir` property starts with the token `{DPIPE_HOME}` then this token is replaced with the value found in the `DPIPE_HOME` environment variable.

The `outputDir` property defaults to the value:

```
{DPIPE_HOME}/data/ImportData/Process_Insight
```

You should not need to modify this property. If you do modify this property then you must also reconfigure the `BPIDP_collect_*.teel` files to also point to this directory.

CSF Field Separator

By default, when the HPBPI Scraper writes its data records to CSF files, the field separator is the tab character.

If you want to change the CSF field separator character used by the Scraper then you set the property `separator`.

You should not need to modify the `separator` property. If you do modify the `separator` property then you must also modify the `FileFormat` property in all the `BPIDP_collect_*.teel` files.

CSF File Base Names

The HPBPI Scraper has a set of properties that allow you to specify the file names to use when creating the CSF output files. These properties are as follows:

- `flowOutFileBase`

  Defaults to `BPIDP_FLOWS_`.

- `flowInstOutFileBase`

  Defaults to `BPIDP_FLOWINST_`.

- `nodeOutFileBase`

  Defaults to `BPIDP_NODES_`.

- `nodeInstOutFileBase`

  Defaults to `BPIDP_NODEINST_`.

- `thresholdOutFileBase`

  Defaults to `BPIDP_THRESHOLDS_`.

- `metricValueOutFileBase`

  Defaults to `BPIDP_METRICVALUES_`.

- `metricStatsOutFileBase`

  Defaults to `BPIDP_METRICSTATS_`.

- `metricAlertOutFileBase`

  Defaults to `BPIDP_METRICALERTS_`.

You should not need modify these properties. If you do modify these properties then you must also modify the `SourceDirectory` property in all the `BPIDP_collect_*.teel` files.

These file names are used in conjunction with the `outputDir` property to create the CSF files. Refer to CSF Output Directory on page 67 for details of the `outputDir` property.

Troubleshooting the HPBPI Scraper

When troubleshooting the HPBPI Scraper, the first thing to do is to check through the log files. The default logging level is `INFO` and if any errors occur then these are logged in the log file as well.

Let's consider some of the ways to troubleshoot the HPBPI Scraper and some of the possible scenarios that might occur.

Debug Logging

If you are having trouble with the HPBPI Scraper then you can turn up the log level to `DEBUG`. Refer to Logging on page 56 for details of how to enable `DEBUG` level logging.

`DEBUG` level logging logs each SQL statement so you can see exactly what is being executed.

Testing the Scraper in Parallel

Sometimes you may have an issue where you want to re-run the Scraper however you do not want to affect the currently running system.

You can run a parallel version of the Scraper on a customer's PI server in such a way that it has no affect on any currently scheduled/running Scraper.

Here are the steps involved to run a parallel Scraper:

1. Create a new directory on the PI server

   This directory can be anywhere. For this example, suppose you create the directory: `/tmp/test`

2. Copy the Scraper property files into this new directory.

   Copy the `bpi_dpipe.properties` and `bpi_dpipe_history.properties` files from the *`DPIPE_HOME`*`/lib` directory and place these copies in your `/tmp/test` directory.

3. Output CSF files to a new output directory

   Edit your copy of the `bpi_dpipe.properties` file (the copy in your `/tmp/test` directory) and set the `outputDir` property to point to a new directory. For example, a new directory called `/tmp/CSFoutput`.

   You can set other things as well if you require, such as setting the log level to `DEBUG`.

4. Set any previous scrape times

   You may want to edit your copy of the `bpi_dipe_history.properties` file and alter the last scrape time of some of the instance settings. For example, if you are wanting to investigate what node instance records are retrieved when the previous time stamp is set to a particular value, then you can edit the `lastTimeStamp_Nodeinst` property and set it to the time stamp that you require for your test. If you want to test a scrape from the epoch time then you can remove (or comment-out) the appropriate last time stamp property.

5. Run the Scraper

   You need to run the Scraper and tell it to use your new test directory that contains your new `bpi_dpipe*` property files. You do this using the `-cfg` command line property.

   For example:

   — To run the Scraper in `definitions` mode and tell it to use the `/tmp/test` directory for its configuration, you issue the following command:

   ```
   DPIPE_HOME/bin/bpi_dpipe -mode definitions -cfg /tmp/test
   ```

   — To run the Scraper in `instances` mode and tell it to use the `/tmp/test` directory for its configuration, you issue the following command:

   ```
   DPIPE_HOME/bin/bpi_dpipe -mode instances -cfg /tmp/test
   ```

   where *DPIPE_HOME* is the directory in which PI is installed.

   The Scraper then runs using the `/tmp/test/bpi_dpipe*` property files to read its configuration details. Because the `outputDir` property has been set in the `bpi_dpipe.properties` file, all output from this run of the Scraper is output to files in the `/tmp/CSFoutput` directory. You are then able to look through these CSF files and see if the output matches your expectation.

By being able to maintain a separate configuration directory and output directory, you are able to run your own test instance of the Scraper. This allows you to examine the data retrieved for given time periods and compare this with the values you were expecting.

Running an instance of the Scraper in parallel as shown above has no affect on any current Scraper instance, however, you should try to avoid running your parallel instance of the Scraper at the same time as another Scraper instance is actually running. Having two instances of the Scraper running at the same time can have an impact on the performance of the HPBPI database.

JVM Exception – Metric Statistics Duplication

It is possible that the HPBPI Scraper runs out of memory when retrieving the metric statistics.

The problem is due to the fact that the Scraper has been written to filter out any duplicate metric statistics records that may appear within the HPBPI database, and this filtering requires the Scraper to maintain an in-memory list of metric statistics IDs.

If your Scraper is unable to retrieve all metric statistics because of an out-of-memory problem then you can try one or more of the following ideas:

- Retrieve less instances

  If the problem occurs when you are running the Scraper for the first time and you have set an epoch start time, you could look at setting the epoch start time to be a date that is not as old. Refer to Epoch Start Date on page 58 for details. You would also need to check the `bpi_dpipe_history.properties` file and see if the `lastTimeStamp_MetricStats` is set. If this property is set then there is no point setting the epoch start time as it will be ignored. If there is a `lastTimeStamp_MetricStats` entry in the history properties file then you simply need to rerun the Scraper to have it continue from where it left off.

  If the problem occurs during a run of the Scraper and it is not the first time you have run the Scraper, then you might want to consider scheduling the Scraper to run (in `instances` mode) more frequently. This would mean that each time the Scraper retrieves instance records there are less to retrieve.

- Sort out the Metric duplication within the HPBPI database

  The metric statistic duplication occurs within HPBPI when an HPBPI metric has been "updated" rather than "replaced". You could consider going into HPBPI and replacing the offending metric definition. However, this would remove the history for this metric.

- Disable the duplication check

  You can edit the `bpi_dpipe.properties` file and set the `duplicateRecordCheck` property to `false`. This disables the metric statistics duplication check. This does however mean that any duplicated metric statistics are retrieved and brought across to the PI database. These duplication can cause the metric statistics reports to fail when trying to render the graphs. You could consider writing SQL scripts to manipulate the `R_BPI_METRICSTATSDATA` table and remove any duplicate entries.

Metric Group Values Truncated at 180 Characters

Within HPBPI, a metric's group value attribute is defined as being a string up to 256 characters in length. The HPBPI Scraper is written to truncate the metric group value attribute at 180 characters in length.

This means that if your HPBPI database does contain metric groups where their group values do not differ within the first 180 characters, then these group values will come across to PI and be reported as if they were within the same metric group value. This could lead to some of the graphs within the metric reports being unable to display due to duplicate data for the same time period. Refer to Metric Group Value Truncation on page 64 for more details.

JDBC Fetch Size and Performance

When the HPBPI Scraper is retrieving instance data, the Scraper tells the JDBC subsystem to bring the physical data across in batches of up to 20000 records. The Scraper still issues a single select statement when it asks to retrieve a set of data, however the underlying JDBC subsystem pages the physical transmission of data records.

For some HPBPI installations, you might find that you can increase the performance of the instance retrievals by increasing this fetch size limit. Refer to JDBC ResultSet Fetch Size on page 64 for details of how to alter the fetch size limit.

Database Deadlocks

When running in `instances` mode, the HPBPI Scraper issues select statements against the HPBPI database that can retrieve up to many thousands of rows of data. Processing these thousands of data rows means that a read-lock can be in place for a considerable amount of time.

If the HPBPI database is MSSQL then a long duration read-lock can lead to deadlocks occurring between the Scraper's select statement and the HPBPI server. If the database detects a deadlock situation then the Scraper's select statement is the most likely candidate to be terminated.

If the Scraper's select statement is terminated due to a deadlock, an SQL exception is written to the Scraper logs, logged as an `ERROR`. You just need to re-run the Scraper and the next time it issues the select statement the Scraper continues from where it was interrupted.

4 Reports

The HPBPI Report Pack provides a set of out-of-the-box reports.

There are reports for the following areas of HPBPI:

- Flows
- Nodes
- Thresholds
- Metric Alerts
- Metric Statistics
- Metric Values

There are reports that show near real time views of the data, as well as reports that provide hourly, daily, weekly and monthly summaries.

This chapter looks at each of the areas of reporting available.

Flows

The following flow reports are available:

- Overall flow status – near real time

  Shows data such as the current active flow instance count, the current highest metric alert against a flow and the extent of any IT impact on the business.

- Flow instance summaries – Hourly/Daily/Weekly/Monthly

  Shows the flow instance completion rate, initiation rate, average duration and total weight for all flow instances.

- Flow instance listings showing alert levels

  Lists completed flow instance details including the business data attributes, and the highest metric alert level that was thrown for the flow instance (if any). The user can filter this report by alert level and/or business data values.

- Flow instance alerts summaries – Hourly/Daily/Weekly/Monthly

  Shows how many flow instances completed having thrown an alert. The report then compares these with the total number of flow instances to show the percentage of flow instances that threw alerts.

- Flow instance listings showing performance against deadlines

  Lists completed flow instances that had a deadline metric associated with them. The report lists the flow instance details, including the business data, and how overdue/ahead the instance was against its deadline. The user can filter this report by deadline time and/or business data values.

- Flow instance deadline summaries – Hourly/Daily/Weekly/Monthly

  Shows how the flow instances have performed against their deadlines.

Let's now consider some examples of these reports.

Overall Flow Status – Near Real Time

Figure 8 on page 75 shows an example of the near real time flows report.

This report enables you to see both the current status of your flow as well as its status over time. The report shows the active flow instance count, the highest metric alert level, and the extent of any IT impact on the business.

The report file name is `bpi_flows_NRT.rep`.

The PI data tables used in this report are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `R_BPI_FLOWDATA`

  For all the graphs.

The datapipe that provides the data for this report is the `BPIDP_FLOWS` datapipe.

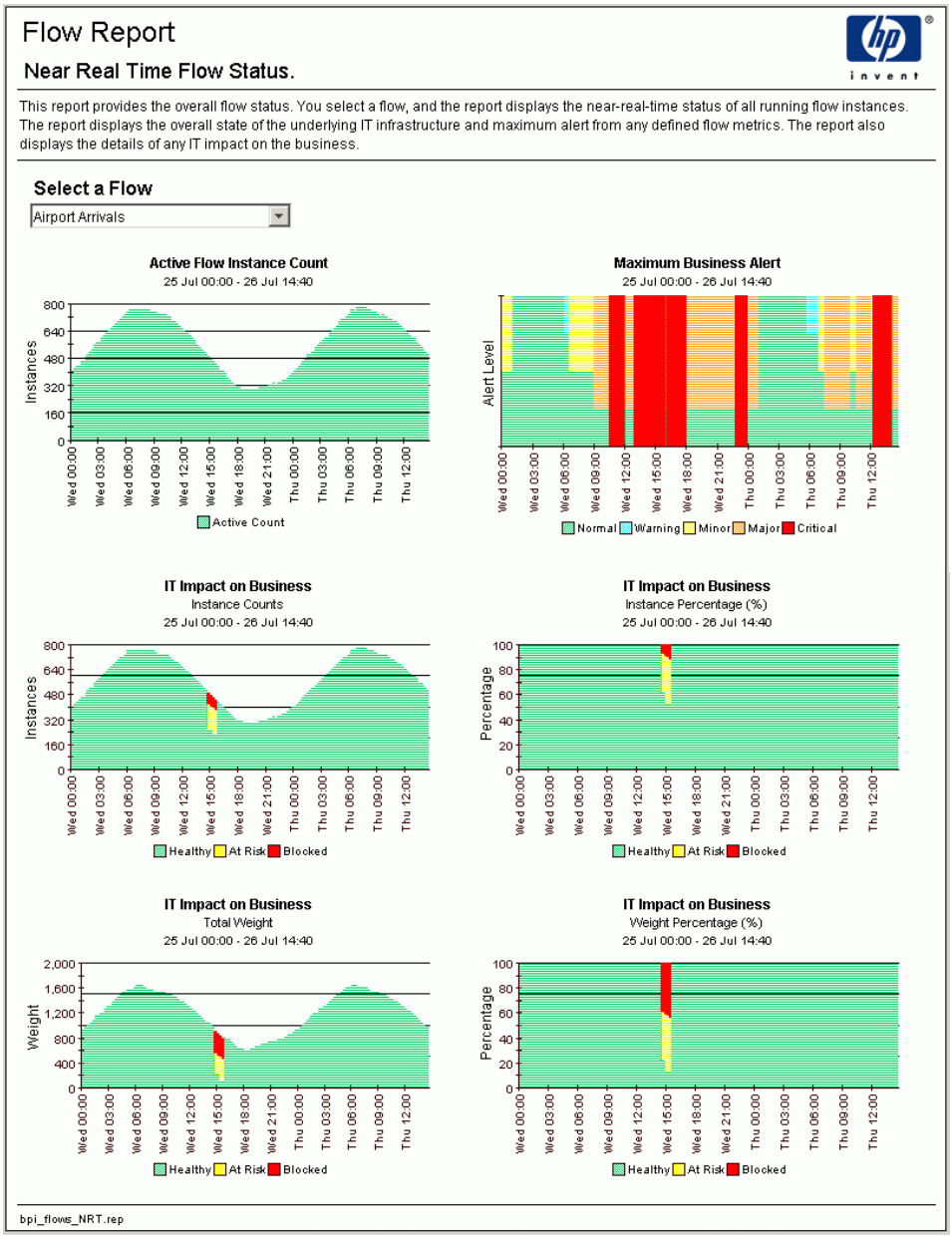
The example report shown in Figure 8 on page 75 has the following points of interest:

- You can see that the maximum number of flow instances per day is just short of 800.

- There was an IT impact on the business and this happened on wednesday at around 3pm.

  You can see that this IT impact caused up to 10% of the flow instances to become blocked, and this amounted to almost 50% of the weight of all flow instances at that time.
- During the hours of 3am and 6am no metric alerts are occurring.
- During the rest of each day the metric alert level varies between warning and critical.

Figure 8 Overall Flow Status Report



Flow Instance Summaries – Hourly/Daily/Weekly/Monthly

Figure 9 on page 77 shows an example of the flow instance summary report. The report shown is an hourly summary. There are also flow instance reports for Daily, Weekly and Monthly summaries.

These summary reports enable you to see the flow instance completion rates, initiation rates, average durations and total weight of your flow instances.

The report file names are as follows:

- `bpi_flows_hourly.rep` for the hourly summary report.
- `bpi_flows_daily.rep` for the daily summary report.
- `bpi_flows_weekly.rep` for the weekly summary report.
- `bpi_flows_monthly.rep` for the monthly summary report.

The PI data tables used in these reports are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `K_BPI_FLOWINSTSTARTTIMES/S*_BPI_FLOWINSTSTARTTIMES`

  For the graph of the initiation rates.

  The initiation rate summaries are derived from the view `RV_BPI_FLOWINSTSTARTTIMES` which provides a view of the `R_BPI_FLOWINSTANCEDATA` table where the ta_period is set to the start time of each flow instance.

- `K_BPI_FLOWINSTANCEDATA/S*_BPI_FLOWINSTANCEDATA`

  For the graphs of the completion rates, average duration and total weight.

where `S*` is either `SH`, `SD`, `SW`, or `SM` depending on whether the report is an hourly, daily, weekly or monthly report.

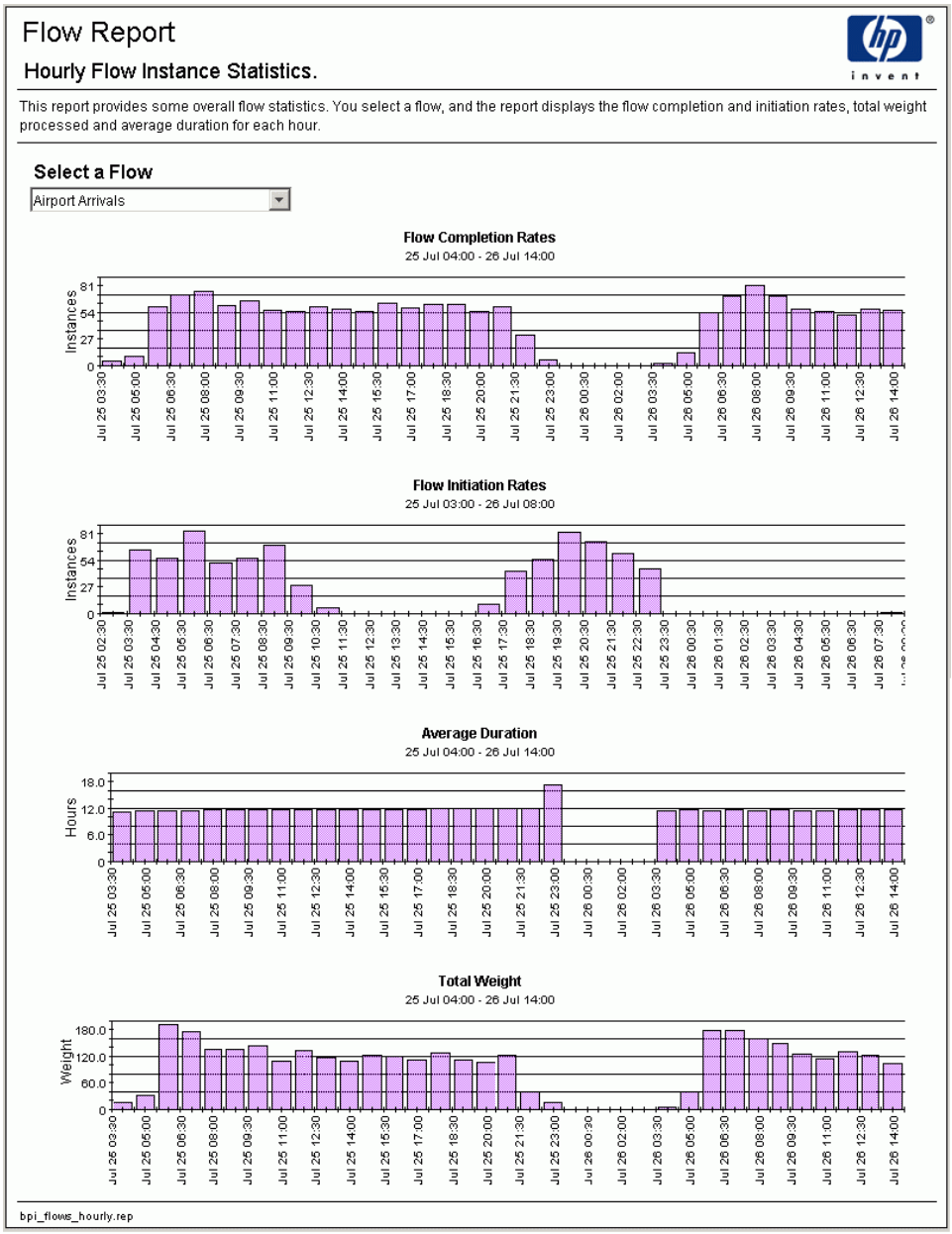
The datapipe that provides the data for these reports is the `BPIDP_FLOWINST` datapipe.

The trends used for these reports are as follows:

- `BPIDP_*_FLOWINSTANCE.sum`
- `BPIDP_*_FLOWINSTSTARTTIMES.sum`

where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

Figure 9 Flow Instance Summary Report



Flow Instance Listings Showing Alert Levels

Figure 10 on page 79 shows an example of the flow instance listing report.

This report lists all recently completed flow instances, showing their business data attributes. If a flow instance threw a metric alert then this report displays the highest alert thrown by that flow instance. The user can filter this report by alert level and/or business data values.

The report file name is `bpi_flowInstanceReport.rep`.

The PI data tables used in this report are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `R_BPI_FLOWINSTANCEDATA`

  For the lists of available business data property values (`dProp*`).

- `RV_BPI_FLOWINSTALERTS`

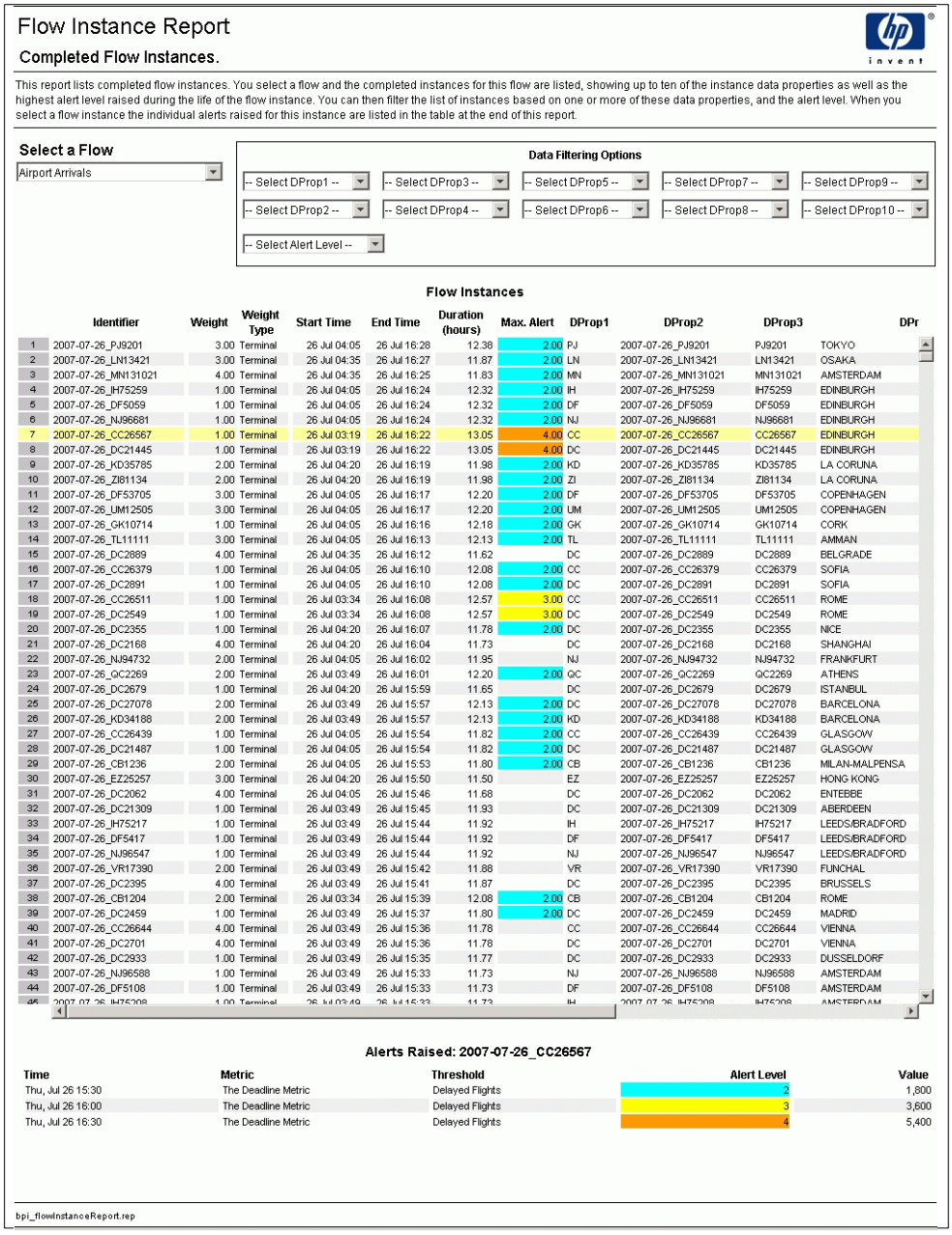
  This is a view that links the two tables `R_BPI_FLOWINSTANCEDATA` and `R_BPI_METRICALERTDATA`. This enables the graph to show the flow instance details with the highest metric alert for each instance.

- `R_BPI_METRICALERTDATA`

  For the list of metric alerts that are displayed at the bottom of the report, when you click on a flow instance

The datapipes that provide the data for this report are the `BPIDP_FLOWINST` and `BPIDP_METRICALERTS` datapipe.

Figure 10 Flow Instance Listing



Flow Instance Report

Completed Flow Instances.

This report lists completed flow instances. You select a flow and the completed instances for this flow are listed, showing up to ten of the instance data properties as well as the highest alert level raised during the life of the flow instance. You can then filter the list of instances based on one or more of these data properties, and the alert level. When you select a flow instance the individual alerts raised for this instance are listed in the table at the end of this report.

Select a Flow: Airport Arrivals

Data Filtering Options: -- Select DProp1 -- | -- Select DProp2 -- | -- Select DProp3 -- | -- Select DProp4 -- | -- Select DProp5 -- | -- Select DProp6 -- | -- Select DProp7 -- | -- Select DProp8 -- | -- Select DProp9 -- | -- Select DProp10 -- | -- Select Alert Level --

Flow Instances

	Identifier	Weight	Weight Type	Start Time	End Time	Duration (hours)	Max. Alert	DProp1	DProp2	DProp3	DPr
1	2007-07-26_PJ9201	3.00	Terminal	26 Jul 04:05	26 Jul 16:28	12.38	2.00	PJ	2007-07-26_PJ9201	PJ9201	TOKYO
2	2007-07-26_LN13421	3.00	Terminal	26 Jul 04:35	26 Jul 16:27	11.87	2.00	LN	2007-07-26_LN13421	LN13421	OSAKA
3	2007-07-26_MN131021	4.00	Terminal	26 Jul 04:35	26 Jul 16:25	11.83	2.00	MN	2007-07-26_MN131021	MN131021	AMSTERDAM
4	2007-07-26_IH75259	1.00	Terminal	26 Jul 04:05	26 Jul 16:24	12.32	2.00	IH	2007-07-26_IH75259	IH75259	EDINBURGH
5	2007-07-26_DF5059	1.00	Terminal	26 Jul 04:05	26 Jul 16:24	12.32	2.00	DF	2007-07-26_DF5059	DF5059	EDINBURGH
6	2007-07-26_NJ96681	1.00	Terminal	26 Jul 04:05	26 Jul 16:24	12.32	2.00	NJ	2007-07-26_NJ96681	NJ96681	EDINBURGH
7	2007-07-26_CC26567	1.00	Terminal	26 Jul 03:19	26 Jul 16:22	13.05	4.00	CC	2007-07-26_CC26567	CC26567	EDINBURGH
8	2007-07-26_DC21445	1.00	Terminal	26 Jul 03:19	26 Jul 16:22	13.05	4.00	DC	2007-07-26_DC21445	DC21445	EDINBURGH
9	2007-07-26_KD35785	2.00	Terminal	26 Jul 04:20	26 Jul 16:19	11.98	2.00	KD	2007-07-26_KD35785	KD35785	LA CORUNA
10	2007-07-26_ZI81134	2.00	Terminal	26 Jul 04:20	26 Jul 16:19	11.98	2.00	ZI	2007-07-26_ZI81134	ZI81134	LA CORUNA
11	2007-07-26_DF53705	3.00	Terminal	26 Jul 04:05	26 Jul 16:17	12.20	2.00	DF	2007-07-26_DF53705	DF53705	COPENHAGEN
12	2007-07-26_UM12505	3.00	Terminal	26 Jul 04:05	26 Jul 16:17	12.20	2.00	UM	2007-07-26_UM12505	UM12505	COPENHAGEN
13	2007-07-26_GK10714	1.00	Terminal	26 Jul 04:05	26 Jul 16:16	12.18	2.00	GK	2007-07-26_GK10714	GK10714	CORK
14	2007-07-26_TL11111	3.00	Terminal	26 Jul 04:05	26 Jul 16:13	12.13	2.00	TL	2007-07-26_TL11111	TL11111	AMMAN
15	2007-07-26_DC2889	4.00	Terminal	26 Jul 04:35	26 Jul 16:12	11.62		DC	2007-07-26_DC2889	DC2889	BELGRADE
16	2007-07-26_CC26379	1.00	Terminal	26 Jul 04:05	26 Jul 16:10	12.08	2.00	CC	2007-07-26_CC26379	CC26379	SOFIA
17	2007-07-26_DC2891	1.00	Terminal	26 Jul 04:05	26 Jul 16:10	12.08	2.00	DC	2007-07-26_DC2891	DC2891	SOFIA
18	2007-07-26_CC26511	1.00	Terminal	26 Jul 03:34	26 Jul 16:08	12.57	3.00	CC	2007-07-26_CC26511	CC26511	ROME
19	2007-07-26_DC2549	1.00	Terminal	26 Jul 03:34	26 Jul 16:08	12.57	3.00	DC	2007-07-26_DC2549	DC2549	ROME
20	2007-07-26_DC2355	1.00	Terminal	26 Jul 04:20	26 Jul 16:07	11.78	2.00	DC	2007-07-26_DC2355	DC2355	NICE
21	2007-07-26_DC2168	4.00	Terminal	26 Jul 04:20	26 Jul 16:04	11.73		DC	2007-07-26_DC2168	DC2168	SHANGHAI
22	2007-07-26_NJ94732	2.00	Terminal	26 Jul 04:05	26 Jul 16:02	11.95		NJ	2007-07-26_NJ94732	NJ94732	FRANKFURT
23	2007-07-26_QC2269	2.00	Terminal	26 Jul 03:49	26 Jul 16:01	12.20	2.00	QC	2007-07-26_QC2269	QC2269	ATHENS
24	2007-07-26_DC2679	1.00	Terminal	26 Jul 04:20	26 Jul 15:59	11.65		DC	2007-07-26_DC2679	DC2679	ISTANBUL
25	2007-07-26_DC27078	2.00	Terminal	26 Jul 03:49	26 Jul 15:57	12.13	2.00	DC	2007-07-26_DC27078	DC27078	BARCELONA
26	2007-07-26_KD34188	2.00	Terminal	26 Jul 03:49	26 Jul 15:57	12.13	2.00	KD	2007-07-26_KD34188	KD34188	BARCELONA
27	2007-07-26_CC26439	1.00	Terminal	26 Jul 04:05	26 Jul 15:54	11.82	2.00	CC	2007-07-26_CC26439	CC26439	GLASGOW
28	2007-07-26_DC21487	1.00	Terminal	26 Jul 04:05	26 Jul 15:54	11.82	2.00	DC	2007-07-26_DC21487	DC21487	GLASGOW
29	2007-07-26_CB1236	2.00	Terminal	26 Jul 04:05	26 Jul 15:53	11.80	2.00	CB	2007-07-26_CB1236	CB1236	MILAN-MALPENSA
30	2007-07-26_EZ25257	3.00	Terminal	26 Jul 04:20	26 Jul 15:50	11.50		EZ	2007-07-26_EZ25257	EZ25257	HONG KONG
31	2007-07-26_DC2062	4.00	Terminal	26 Jul 04:05	26 Jul 15:46	11.68		DC	2007-07-26_DC2062	DC2062	ENTEBBE
32	2007-07-26_DC21309	1.00	Terminal	26 Jul 03:49	26 Jul 15:45	11.93		DC	2007-07-26_DC21309	DC21309	ABERDEEN
33	2007-07-26_IH75217	1.00	Terminal	26 Jul 03:49	26 Jul 15:44	11.92		IH	2007-07-26_IH75217	IH75217	LEEDS/BRADFORD
34	2007-07-26_DF5417	1.00	Terminal	26 Jul 03:49	26 Jul 15:44	11.92		DF	2007-07-26_DF5417	DF5417	LEEDS/BRADFORD
35	2007-07-26_NJ96547	1.00	Terminal	26 Jul 03:49	26 Jul 15:44	11.92		NJ	2007-07-26_NJ96547	NJ96547	LEEDS/BRADFORD
36	2007-07-26_VR17390	2.00	Terminal	26 Jul 03:49	26 Jul 15:42	11.88		VR	2007-07-26_VR17390	VR17390	FUNCHAL
37	2007-07-26_DC2395	4.00	Terminal	26 Jul 03:49	26 Jul 15:41	11.87		DC	2007-07-26_DC2395	DC2395	BRUSSELS
38	2007-07-26_CB1204	2.00	Terminal	26 Jul 03:34	26 Jul 15:39	12.08	2.00	CB	2007-07-26_CB1204	CB1204	ROME
39	2007-07-26_DC2459	1.00	Terminal	26 Jul 03:49	26 Jul 15:37	11.80	2.00	DC	2007-07-26_DC2459	DC2459	MADRID
40	2007-07-26_CC26644	4.00	Terminal	26 Jul 03:49	26 Jul 15:36	11.78		CC	2007-07-26_CC26644	CC26644	VIENNA
41	2007-07-26_DC2701	4.00	Terminal	26 Jul 03:49	26 Jul 15:36	11.78		DC	2007-07-26_DC2701	DC2701	VIENNA
42	2007-07-26_DC2933	1.00	Terminal	26 Jul 03:49	26 Jul 15:35	11.77		DC	2007-07-26_DC2933	DC2933	DUSSELDORF
43	2007-07-26_NJ96588	1.00	Terminal	26 Jul 03:49	26 Jul 15:33	11.73		NJ	2007-07-26_NJ96588	NJ96588	AMSTERDAM
44	2007-07-26_DF5108	1.00	Terminal	26 Jul 03:49	26 Jul 15:33	11.73		DF	2007-07-26_DF5108	DF5108	AMSTERDAM
45	2007-07-26_IH75208	1.00	Terminal	26 Jul 03:49	26 Jul 15:33	11.73		IH	2007-07-26_IH75208	IH75208	AMSTERDAM

Alerts Raised: 2007-07-26_CC26567

Time	Metric	Threshold	Alert Level	Value
Thu, Jul 26 15:30	The Deadline Metric	Delayed Flights	2	1,800
Thu, Jul 26 16:00	The Deadline Metric	Delayed Flights	3	3,600
Thu, Jul 26 16:30	The Deadline Metric	Delayed Flights	4	5,400

bpi_flowInstanceReport.rep

Flow Instance Alerts Summaries – Hourly/Daily/Weekly/Monthly

Figure 11 on page 81 shows an example of the flow instance alert summary report. The report shown is an hourly summary. There are also reports for Daily, Weekly and Monthly summaries.

These summary reports enable you to see the number of flow instances that have thrown metric alerts, as well as this data expressed as a percentage of the total flow instances for that period. The table in the middle of the report captures all the data in tabular form. The data shown in the left four columns of the table (the total counts) are shown in the top graph. The data shown in right four columns of the table (the percentages) are shown in the graph at the bottom of the report.

The report file names are as follows:

- `bpi_flowInstanceAlerts_hourly.rep` for the hourly summary report.
- `bpi_flowInstanceAlerts_daily.rep` for the daily summary.
- `bpi_flowInstanceAlerts_weekly.rep` for the weekly summary.
- `bpi_flowInstanceAlerts_monthly.rep` for the monthly summary.

The PI data tables used in these reports are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `K_BPI_FLOWINSTALERTS/S*_BPI_FLOWINSTALERTS`

  For the graphs and the table of data.

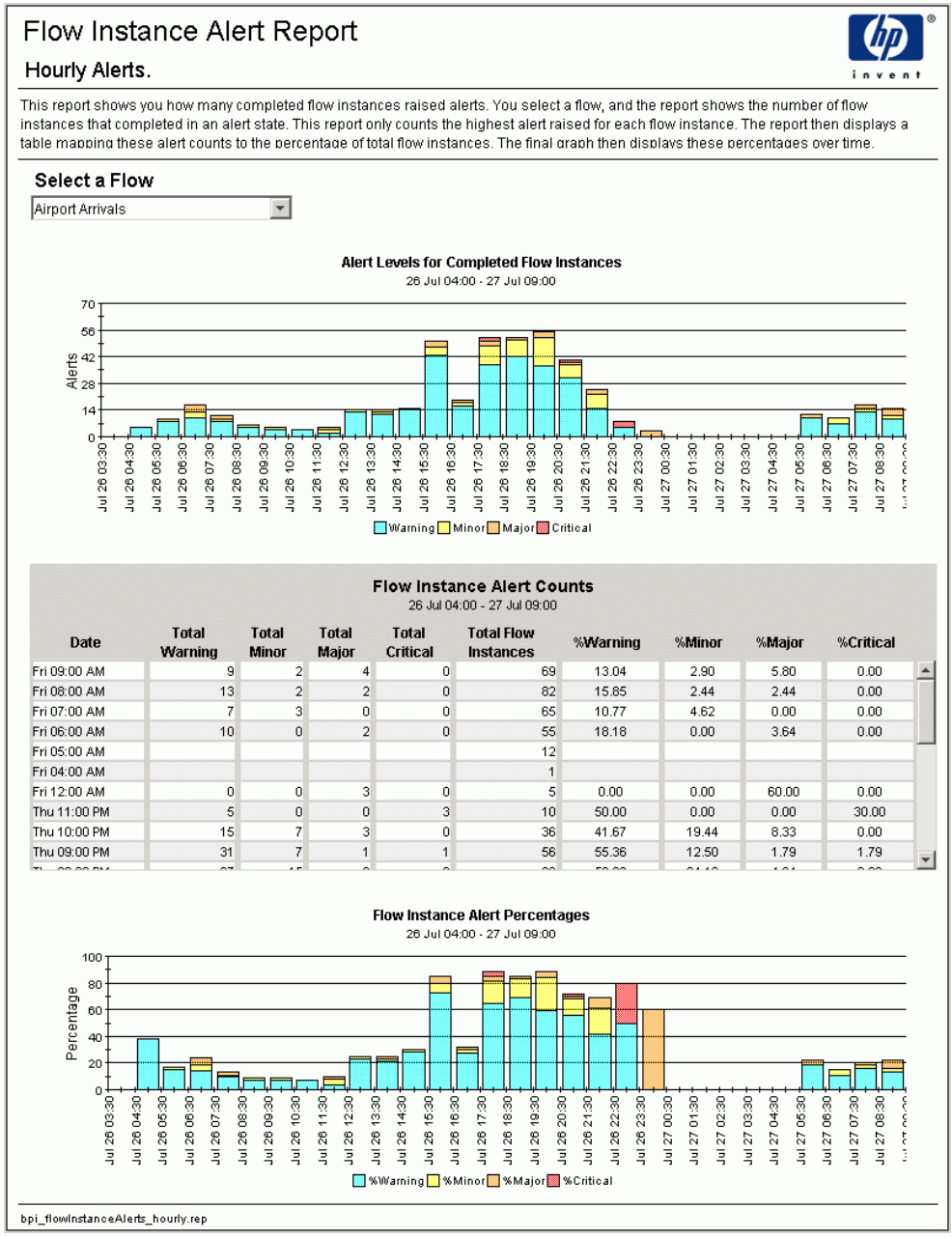
  The flow instance alert summaries are derived from the view `RV_BPI_FLOWINSTALERTS`. This view that links the two tables `R_BPI_FLOWINSTANCEDATA` and `R_BPI_METRICALERTDATA`.

  `S*` is either `SH`, `SD`, `SW`, or `SM` depending on whether the report is an hourly, daily, weekly or monthly report.

The datapipes that provide the data for this report are the `BPIDP_FLOWINST` and `BPIDP_METRICALERTS` datapipe.

The trends used for these reports are named `BPIDP_*_FLOWINSTANCEALERTS.sum` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

Figure 11 Flow Instance Alert Summary Report



Flow Instance Alert Report

Hourly Alerts.

This report shows you how many completed flow instances raised alerts. You select a flow, and the report shows the number of flow instances that completed in an alert state. This report only counts the highest alert raised for each flow instance. The report then displays a table mapping these alert counts to the percentage of total flow instances. The final graph then displays these percentages over time.

Select a Flow

Airport Arrivals

Alert Levels for Completed Flow Instances

26 Jul 04:00 - 27 Jul 09:00

Alerts

Warning Minor Major Critical

Flow Instance Alert Counts

26 Jul 04:00 - 27 Jul 09:00

Date	Total Warning	Total Minor	Total Major	Total Critical	Total Flow Instances	%Warning	%Minor	%Major	%Critical
Fri 09:00 AM	9	2	4	0	69	13.04	2.90	5.80	0.00
Fri 08:00 AM	13	2	2	0	82	15.85	2.44	2.44	0.00
Fri 07:00 AM	7	3	0	0	65	10.77	4.62	0.00	0.00
Fri 06:00 AM	10	0	2	0	55	18.18	0.00	3.64	0.00
Fri 05:00 AM					12				
Fri 04:00 AM					1				
Fri 12:00 AM	0	0	3	0	5	0.00	0.00	60.00	0.00
Thu 11:00 PM	5	0	0	3	10	50.00	0.00	0.00	30.00
Thu 10:00 PM	15	7	3	0	36	41.67	19.44	8.33	0.00
Thu 09:00 PM	31	7	1	1	56	55.36	12.50	1.79	1.79

Flow Instance Alert Percentages

26 Jul 04:00 - 27 Jul 09:00

Percentage

%Warning %Minor %Major %Critical

bpi_flowInstanceAlerts_hourly.rep

Flow Instance Listings Showing Performance Against Deadlines

Figure 12 on page 83 shows an example of the flow instance deadline report.

This report lists all recently completed flow instances that have a deadline metric against them. The report then shows the flow instance details as well as how the instance performed against its deadline. The user can filter this report by business data values and/or by selecting a deadline range. For example, a user can request to see only those instances that were more than 60 minutes overdue.

The report file name is `bpi_flowInstanceDeadlines.rep`.

The PI data tables used in this report are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `RV_BPI_FLOWINSTDEADLINES`

  For the list of available deadline metrics and metric group values, as well as the actual list of flow instances.

  `RV_BPI_FLOWINSTDEADLINES` is a view that links the two tables `R_BPI_FLOWINSTANCEDATA` and `R_BPI_METRICVALUEDATA`. This view joins together all deadline metrics (if any) and their associated flow instances.

- `R_BPI_FLOWINSTANCEDATA`

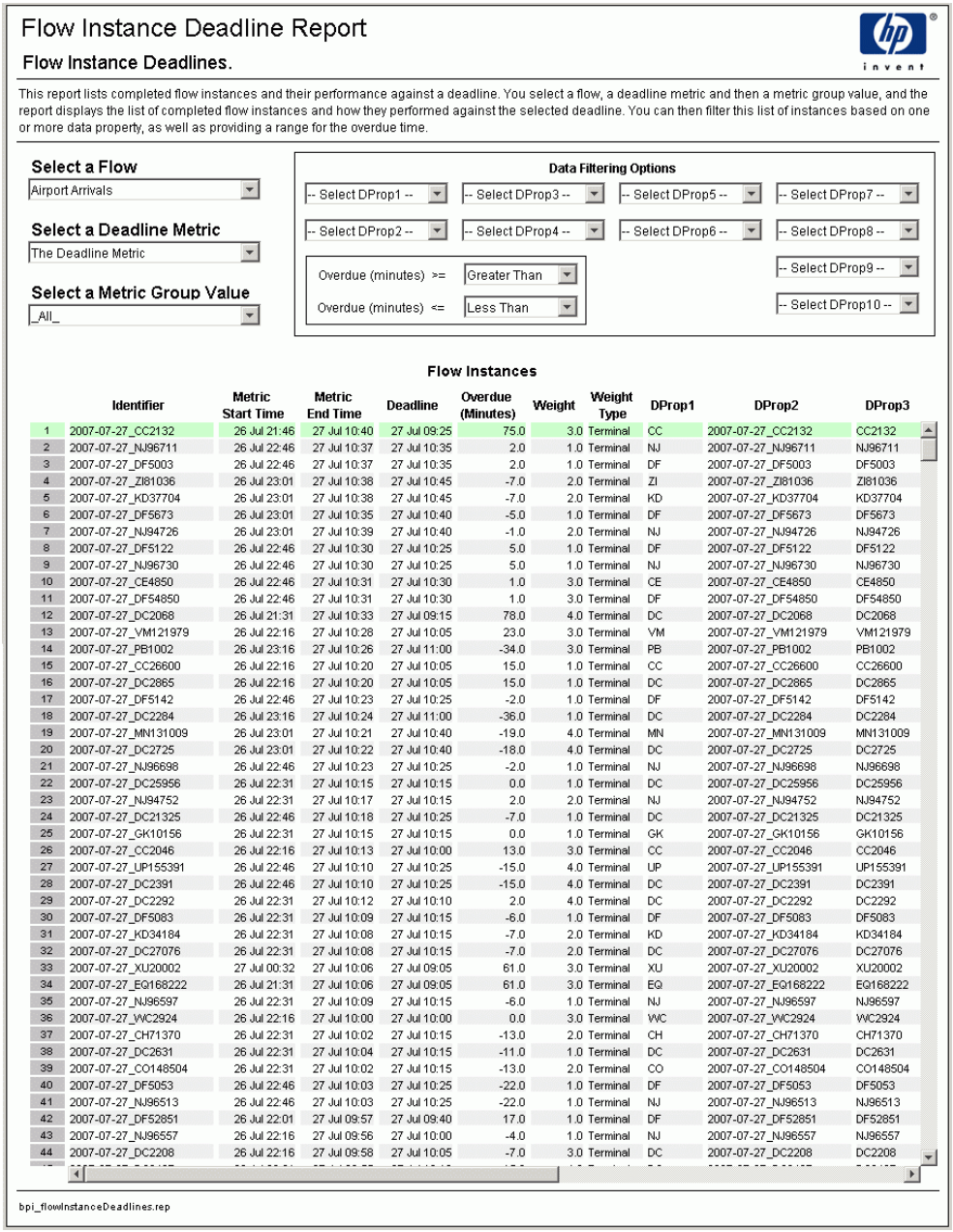
  For the lists of available business data property values (`dProp*`).

- `GB_BPI_DEADLINERANGES`

  For the list of deadline range selection values.

The datapipes that provide the data for this report are the `BPIDP_FLOWINST` and `BPIDP_METRICVALUES` datapipe.

Figure 12 Flow Instance Deadline Report



Flow Instance Deadline Report

Flow Instance Deadlines.

This report lists completed flow instances and their performance against a deadline. You select a flow, a deadline metric and then a metric group value, and the report displays the list of completed flow instances and how they performed against the selected deadline. You can then filter this list of instances based on one or more data property, as well as providing a range for the overdue time.

Select a Flow: Airport Arrivals

Select a Deadline Metric: The Deadline Metric

Select a Metric Group Value: _All_

Data Filtering Options

-- Select DProp1 -- | -- Select DProp3 -- | -- Select DProp5 -- | -- Select DProp7 --

-- Select DProp2 -- | -- Select DProp4 -- | -- Select DProp6 -- | -- Select DProp8 --

Overdue (minutes) >= Greater Than | -- Select DProp9 --

Overdue (minutes) <= Less Than | -- Select DProp10 --

Flow Instances

	Identifier	Metric Start Time	Metric End Time	Deadline	Overdue (Minutes)	Weight	Weight Type	DProp1	DProp2	DProp3
1	2007-07-27_CC2132	26 Jul 21:46	27 Jul 10:40	27 Jul 09:25	75.0	3.0	Terminal	CC	2007-07-27_CC2132	CC2132
2	2007-07-27_NJ96711	26 Jul 22:46	27 Jul 10:37	27 Jul 10:35	2.0	1.0	Terminal	NJ	2007-07-27_NJ96711	NJ96711
3	2007-07-27_DF5003	26 Jul 22:46	27 Jul 10:37	27 Jul 10:35	2.0	1.0	Terminal	DF	2007-07-27_DF5003	DF5003
4	2007-07-27_ZI81036	26 Jul 23:01	27 Jul 10:38	27 Jul 10:45	-7.0	2.0	Terminal	ZI	2007-07-27_ZI81036	ZI81036
5	2007-07-27_KD37704	26 Jul 23:01	27 Jul 10:38	27 Jul 10:45	-7.0	2.0	Terminal	KD	2007-07-27_KD37704	KD37704
6	2007-07-27_DF5673	26 Jul 23:01	27 Jul 10:35	27 Jul 10:40	-5.0	1.0	Terminal	DF	2007-07-27_DF5673	DF5673
7	2007-07-27_NJ94726	26 Jul 23:01	27 Jul 10:39	27 Jul 10:40	-1.0	2.0	Terminal	NJ	2007-07-27_NJ94726	NJ94726
8	2007-07-27_DF5122	26 Jul 22:46	27 Jul 10:30	27 Jul 10:25	5.0	1.0	Terminal	DF	2007-07-27_DF5122	DF5122
9	2007-07-27_NJ96730	26 Jul 22:46	27 Jul 10:30	27 Jul 10:25	5.0	1.0	Terminal	NJ	2007-07-27_NJ96730	NJ96730
10	2007-07-27_CE4850	26 Jul 22:46	27 Jul 10:31	27 Jul 10:30	1.0	3.0	Terminal	CE	2007-07-27_CE4850	CE4850
11	2007-07-27_DF54850	26 Jul 22:46	27 Jul 10:31	27 Jul 10:30	1.0	3.0	Terminal	DF	2007-07-27_DF54850	DF54850
12	2007-07-27_DC2068	26 Jul 21:31	27 Jul 10:33	27 Jul 09:15	78.0	4.0	Terminal	DC	2007-07-27_DC2068	DC2068
13	2007-07-27_VM121979	26 Jul 22:16	27 Jul 10:28	27 Jul 10:05	23.0	3.0	Terminal	VM	2007-07-27_VM121979	VM121979
14	2007-07-27_PB1002	26 Jul 23:16	27 Jul 10:26	27 Jul 11:00	-34.0	3.0	Terminal	PB	2007-07-27_PB1002	PB1002
15	2007-07-27_CC26600	26 Jul 22:16	27 Jul 10:20	27 Jul 10:05	15.0	1.0	Terminal	CC	2007-07-27_CC26600	CC26600
16	2007-07-27_DC2865	26 Jul 22:16	27 Jul 10:20	27 Jul 10:05	15.0	1.0	Terminal	DC	2007-07-27_DC2865	DC2865
17	2007-07-27_DF5142	26 Jul 22:46	27 Jul 10:23	27 Jul 10:25	-2.0	1.0	Terminal	DF	2007-07-27_DF5142	DF5142
18	2007-07-27_DC2284	26 Jul 23:16	27 Jul 10:24	27 Jul 11:00	-36.0	1.0	Terminal	DC	2007-07-27_DC2284	DC2284
19	2007-07-27_MN131009	26 Jul 23:01	27 Jul 10:21	27 Jul 10:40	-19.0	4.0	Terminal	MN	2007-07-27_MN131009	MN131009
20	2007-07-27_DC2725	26 Jul 23:01	27 Jul 10:22	27 Jul 10:40	-18.0	4.0	Terminal	DC	2007-07-27_DC2725	DC2725
21	2007-07-27_NJ96698	26 Jul 22:46	27 Jul 10:23	27 Jul 10:25	-2.0	1.0	Terminal	NJ	2007-07-27_NJ96698	NJ96698
22	2007-07-27_DC25956	26 Jul 22:31	27 Jul 10:15	27 Jul 10:15	0.0	1.0	Terminal	DC	2007-07-27_DC25956	DC25956
23	2007-07-27_NJ94752	26 Jul 22:31	27 Jul 10:17	27 Jul 10:15	2.0	2.0	Terminal	NJ	2007-07-27_NJ94752	NJ94752
24	2007-07-27_DC21325	26 Jul 22:46	27 Jul 10:18	27 Jul 10:25	-7.0	1.0	Terminal	DC	2007-07-27_DC21325	DC21325
25	2007-07-27_GK10156	26 Jul 22:31	27 Jul 10:15	27 Jul 10:15	0.0	1.0	Terminal	GK	2007-07-27_GK10156	GK10156
26	2007-07-27_CC2046	26 Jul 22:16	27 Jul 10:13	27 Jul 10:00	13.0	3.0	Terminal	CC	2007-07-27_CC2046	CC2046
27	2007-07-27_UP155391	26 Jul 22:46	27 Jul 10:10	27 Jul 10:25	-15.0	4.0	Terminal	UP	2007-07-27_UP155391	UP155391
28	2007-07-27_DC2391	26 Jul 22:46	27 Jul 10:10	27 Jul 10:25	-15.0	4.0	Terminal	DC	2007-07-27_DC2391	DC2391
29	2007-07-27_DC2292	26 Jul 22:31	27 Jul 10:12	27 Jul 10:10	2.0	4.0	Terminal	DC	2007-07-27_DC2292	DC2292
30	2007-07-27_DF5083	26 Jul 22:31	27 Jul 10:09	27 Jul 10:15	-6.0	1.0	Terminal	DF	2007-07-27_DF5083	DF5083
31	2007-07-27_KD34184	26 Jul 22:31	27 Jul 10:08	27 Jul 10:15	-7.0	2.0	Terminal	KD	2007-07-27_KD34184	KD34184
32	2007-07-27_DC27076	26 Jul 22:31	27 Jul 10:08	27 Jul 10:15	-7.0	2.0	Terminal	DC	2007-07-27_DC27076	DC27076
33	2007-07-27_XU20002	27 Jul 00:32	27 Jul 10:06	27 Jul 09:05	61.0	3.0	Terminal	XU	2007-07-27_XU20002	XU20002
34	2007-07-27_EQ168222	26 Jul 21:31	27 Jul 10:06	27 Jul 09:05	61.0	3.0	Terminal	EQ	2007-07-27_EQ168222	EQ168222
35	2007-07-27_NJ96597	26 Jul 22:31	27 Jul 10:09	27 Jul 10:15	-6.0	1.0	Terminal	NJ	2007-07-27_NJ96597	NJ96597
36	2007-07-27_WC2924	26 Jul 22:16	27 Jul 10:00	27 Jul 10:00	0.0	3.0	Terminal	WC	2007-07-27_WC2924	WC2924
37	2007-07-27_CH71370	26 Jul 22:31	27 Jul 10:02	27 Jul 10:15	-13.0	2.0	Terminal	CH	2007-07-27_CH71370	CH71370
38	2007-07-27_DC2631	26 Jul 22:31	27 Jul 10:04	27 Jul 10:15	-11.0	1.0	Terminal	DC	2007-07-27_DC2631	DC2631
39	2007-07-27_CO148504	26 Jul 22:31	27 Jul 10:02	27 Jul 10:15	-13.0	2.0	Terminal	CO	2007-07-27_CO148504	CO148504
40	2007-07-27_DF5053	26 Jul 22:46	27 Jul 10:03	27 Jul 10:25	-22.0	1.0	Terminal	DF	2007-07-27_DF5053	DF5053
41	2007-07-27_NJ96513	26 Jul 22:46	27 Jul 10:03	27 Jul 10:25	-22.0	1.0	Terminal	NJ	2007-07-27_NJ96513	NJ96513
42	2007-07-27_DF52851	26 Jul 22:01	27 Jul 09:57	27 Jul 09:40	17.0	1.0	Terminal	DF	2007-07-27_DF52851	DF52851
43	2007-07-27_NJ96557	26 Jul 22:16	27 Jul 09:56	27 Jul 10:00	-4.0	1.0	Terminal	NJ	2007-07-27_NJ96557	NJ96557
44	2007-07-27_DC2208	26 Jul 22:16	27 Jul 09:58	27 Jul 10:05	-7.0	3.0	Terminal	DC	2007-07-27_DC2208	DC2208

bpi_flowInstanceDeadlines.rep

Flow Instance Deadline Summaries – Hourly/Daily/Weekly/Monthly

Figure 13 on page 85 shows an example of the flow instance deadline summary report. The report shown is an hourly summary. There are also reports for Daily, Weekly and Monthly summaries.

These summary reports enable you to see the following:

- The overall performance of the flow against the selected deadline metric.
- The instance counts ahead, on time and overdue for the deadline.
- The average/min/max for all overdue instances.
- The average/min/max for all instances that are ahead of the deadline.

The report file names are `bpi_flowInstanceDeadlines_*.rep` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

The PI data tables used in these reports are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `RV_BPI_FLOWINSTDEADLINES`

  For the list of available deadline metrics and metric group values.

  `RV_BPI_FLOWINSTDEADLINES` is a view that links the two tables `R_BPI_FLOWINSTANCEDATA` and `R_BPI_METRICVALUEDATA`. This view joins together all deadline metrics (if any) and their associated flow instances.

- `K_BPI_FLOWINSTDEADLINES/S*_BPI_FLOWINSTDEADLINES`

  For all the graphs.

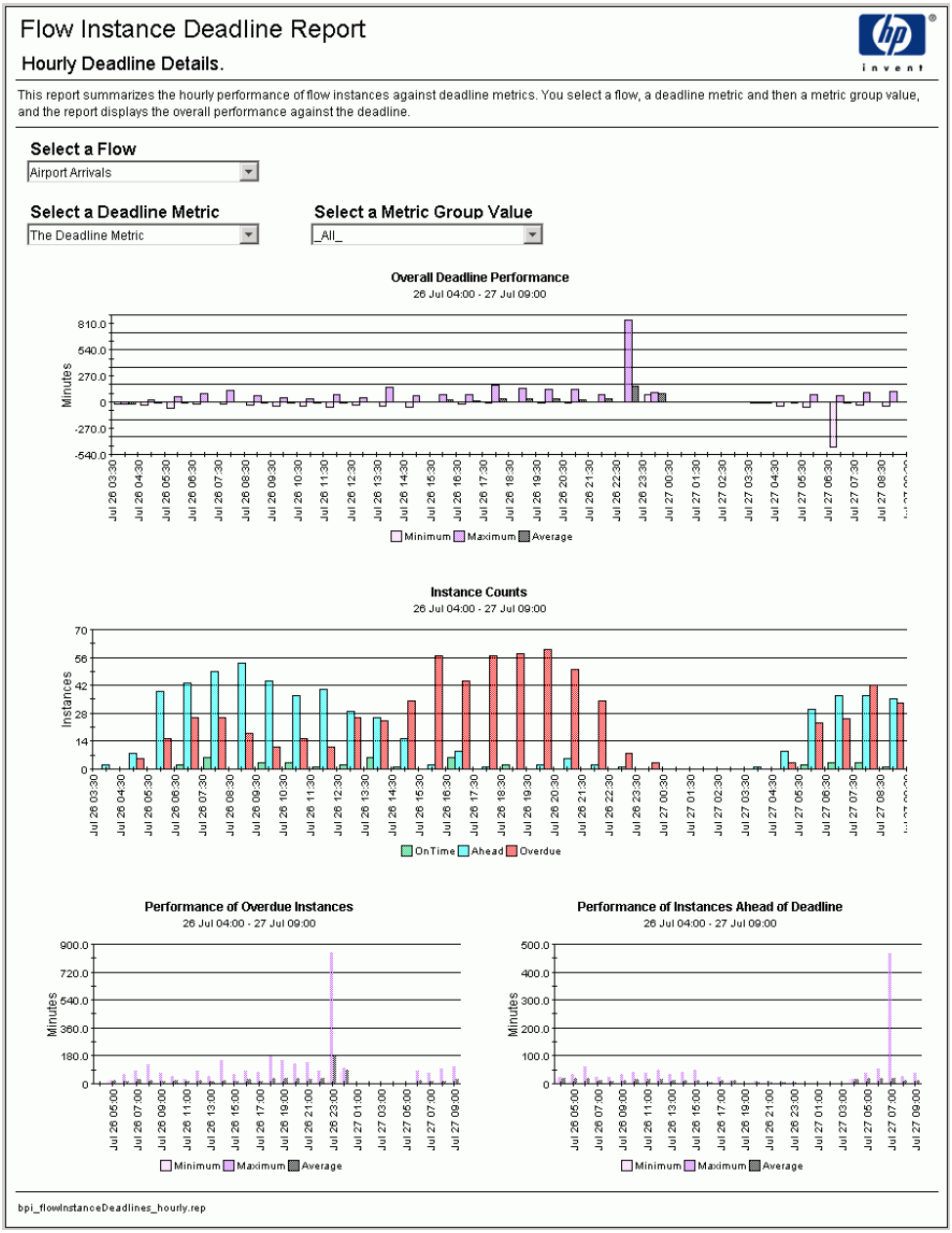
  These deadline summaries are derived from the view `RV_BPI_FLOWINSTDEADLINES`.

  `S*` is either `SH`, `SD`, `SW`, or `SM` depending on whether the report is an hourly, daily, weekly or monthly report.

The datapipes that provide the data for this report are the `BPIDP_FLOWINST` and `BPIDP_METRICVALUES` datapipe.

The trends used for these reports are named `BPIDP_*_FLOWINSTANCEDEADLINES.sum` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

Figure 13 Flow Instance Deadline Summary Report



Nodes

The following node reports are available:

- Overall node status – near real time

  Shows data such as the current active node instance count, node instance throughput and any IT impact on the node.

- Node instance summaries – Hourly/Daily/Weekly/Monthly

  Shows the node instance completion rate, initiation rate and node duration.

Let's now consider some examples of these reports.

Overall Node Status – Near Real Time

Figure 14 on page 87 shows an example of the near real time nodes report.

This report enables you to see both the current status of a selected node as well as its status over time. The report shows the active node instance count, throughput, weight rates, and any IT impact for the node.

The report file name is `bpi_nodes_NRT.rep`.

The PI data tables used in this report are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `K_BPI_NODES`

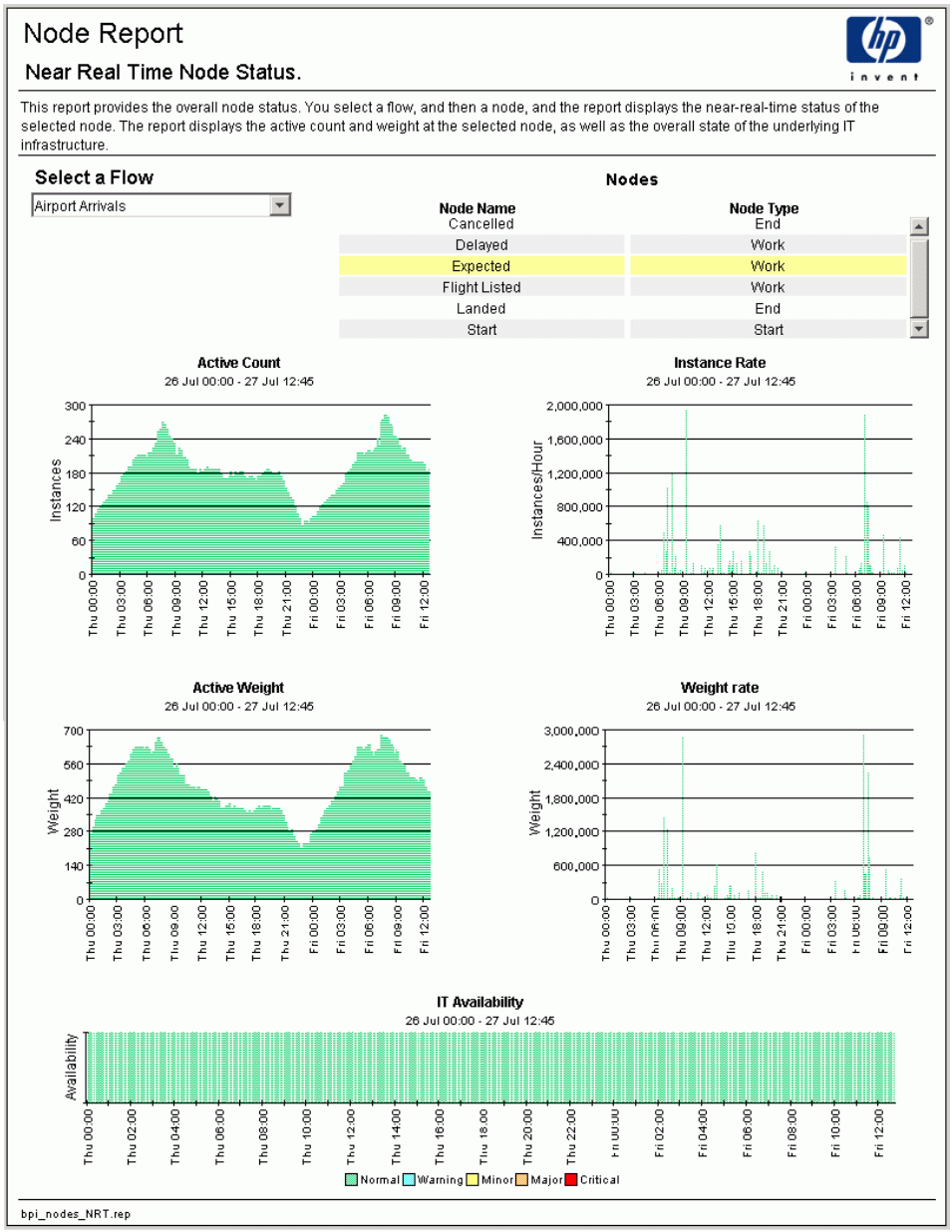
  For the list of node names.

- `R_BPI_NODEDATA`

  For all the graphs.

The datapipe that provides the data for this report is the `BPIDP_NODES` datapipe.

Figure 14 Overall Node Status Report



Node Instance Summaries – Hourly/Daily/Weekly/Monthly

Figure 15 on page 89 shows an example of the node instance summary report. The report shown is an hourly summary. There are also reports for Daily, Weekly and Monthly summaries.

These summary reports enable you to see the node instance completion rates, initiation rates and node durations.

The report file names are `bpi_nodes_*.rep` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

The PI data tables used in these reports are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `K_BPI_NODES`

  For the list of node names.

- `K_BPI_NODEINSTSTARTTIMES/S*_BPI_NODEINSTSTARTTIMES`

  For the graph of the initiation rates.

  The initiation rate summaries are derived from the view `RV_BPI_NODEINSTSTARTTIMES` which provides a view of the `R_BPI_NODEINSTANCEDATA` table where the ta_period is set to the start time of each node instance.

- `K_BPI_NODEINSTANCEDATA/S*_BPI_NODEINSTANCEDATA`

  For the graphs of the completion rates and the duration.

where `S*` is either `SH`, `SD`, `SW`, or `SM` depending on whether the report is an hourly, daily, weekly or monthly report.

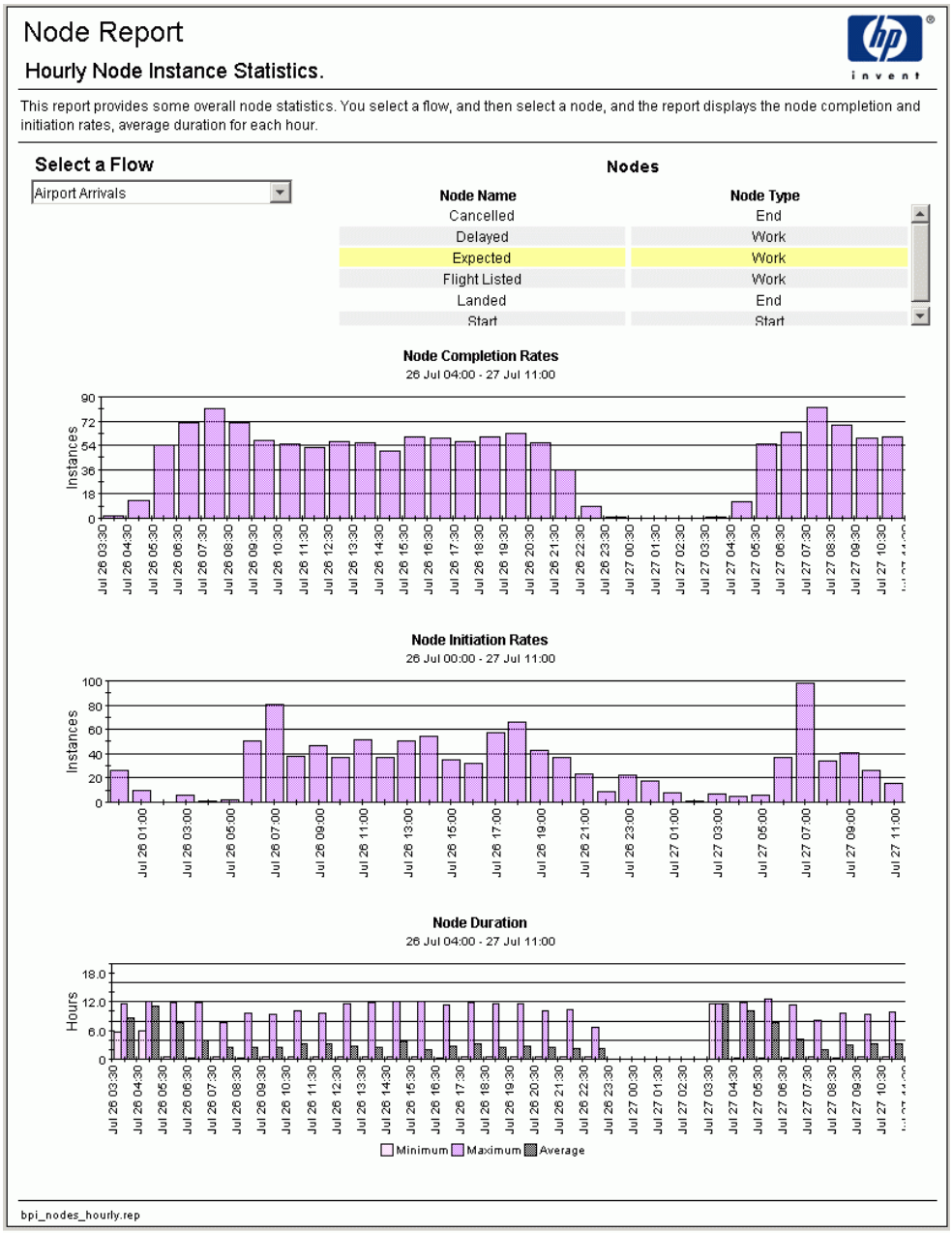
The datapipe that provides the data for these reports is the `BPIDP_NODEINST` datapipe.

The trends used for these reports are as follows:

- `BPIDP_*_NODEINSTANCE.sum`
- `BPIDP_*_NODEINSTSTARTTIMES.sum`

where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

Figure 15 Node Instance Summary Report



Thresholds

The following threshold reports are available:

- Overall threshold status – near real time

  Shows the current threshold alert level.

- Threshold summaries – Hourly/Daily/Weekly/Monthly

  Shows the average, minimum and maximum threshold alert levels for the given time period.

Let's now consider some examples of these reports.

Overall Threshold Status – Near Real Time

Figure 16 on page 91 shows an example of the near real time threshold report.

This report enables you to see both the current alert level for a selected threshold as well as its alert levels over time.

The report file name is `bpi_thresholds_NRT.rep`.

The PI data tables used in this report are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `K_BPI_METRICS`

  For the list of metric names.

- `K_BPI_THRESHOLDS`

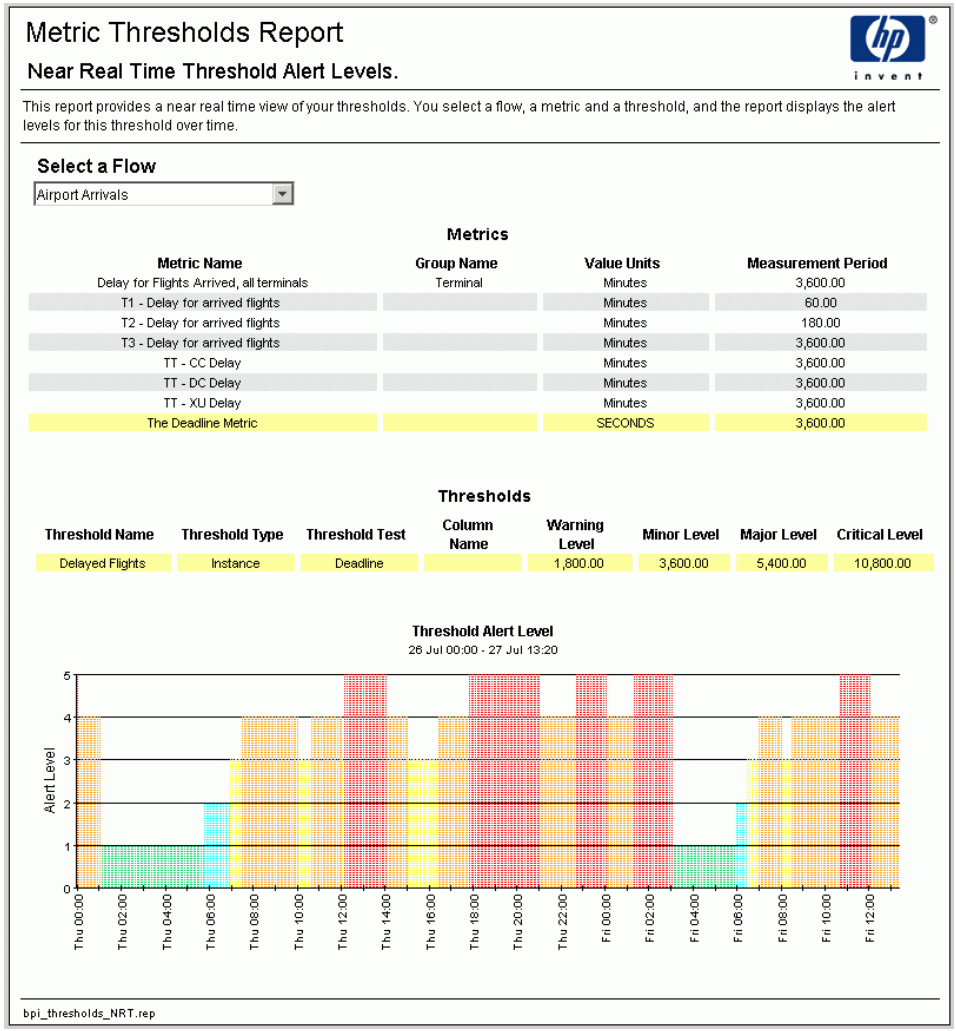
  For the list of threshold names.

- `R_BPI_THRESHOLDDATA`

  For the graph of the alert levels.

The datapipe that provides the data for this report is the `BPIDP_THRESHOLDS` datapipe.

Figure 16 Overall Threshold Status Report



Metric Thresholds Report

Near Real Time Threshold Alert Levels.

This report provides a near real time view of your thresholds. You select a flow, a metric and a threshold, and the report displays the alert levels for this threshold over time.

Select a Flow

Airport Arrivals

Metrics

Metric Name	Group Name	Value Units	Measurement Period
Delay for Flights Arrived, all terminals	Terminal	Minutes	3,600.00
T1 - Delay for arrived flights		Minutes	60.00
T2 - Delay for arrived flights		Minutes	180.00
T3 - Delay for arrived flights		Minutes	3,600.00
TT - CC Delay		Minutes	3,600.00
TT - DC Delay		Minutes	3,600.00
TT - XU Delay		Minutes	3,600.00
The Deadline Metric		SECONDS	3,600.00

Thresholds

Threshold Name	Threshold Type	Threshold Test	Column Name	Warning Level	Minor Level	Major Level	Critical Level
Delayed Flights	Instance	Deadline		1,800.00	3,600.00	5,400.00	10,800.00

Threshold Alert Level

26 Jul 00:00 - 27 Jul 13:20

bpi_thresholds_NRT.rep

Threshold Summaries – Hourly/Daily/Weekly/Monthly

Figure 17 on page 93 shows an example of the threshold summary report. The report shown is an hourly summary. There are also reports for Daily, Weekly and Monthly summaries.

These summary reports enable you to see the average, minimum and maximum threshold alert levels for the given time period.

The report file names are `bpi_thresholds_*.rep` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

The PI data tables used in these reports are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `K_BPI_METRICS`

  For the list of metric names.

- `K_BPI_THRESHOLDS`

  For the list of threshold names.

- `K_BPI_THRESHOLDLEVELS/S*_BPI_THRESHOLDLEVELS`

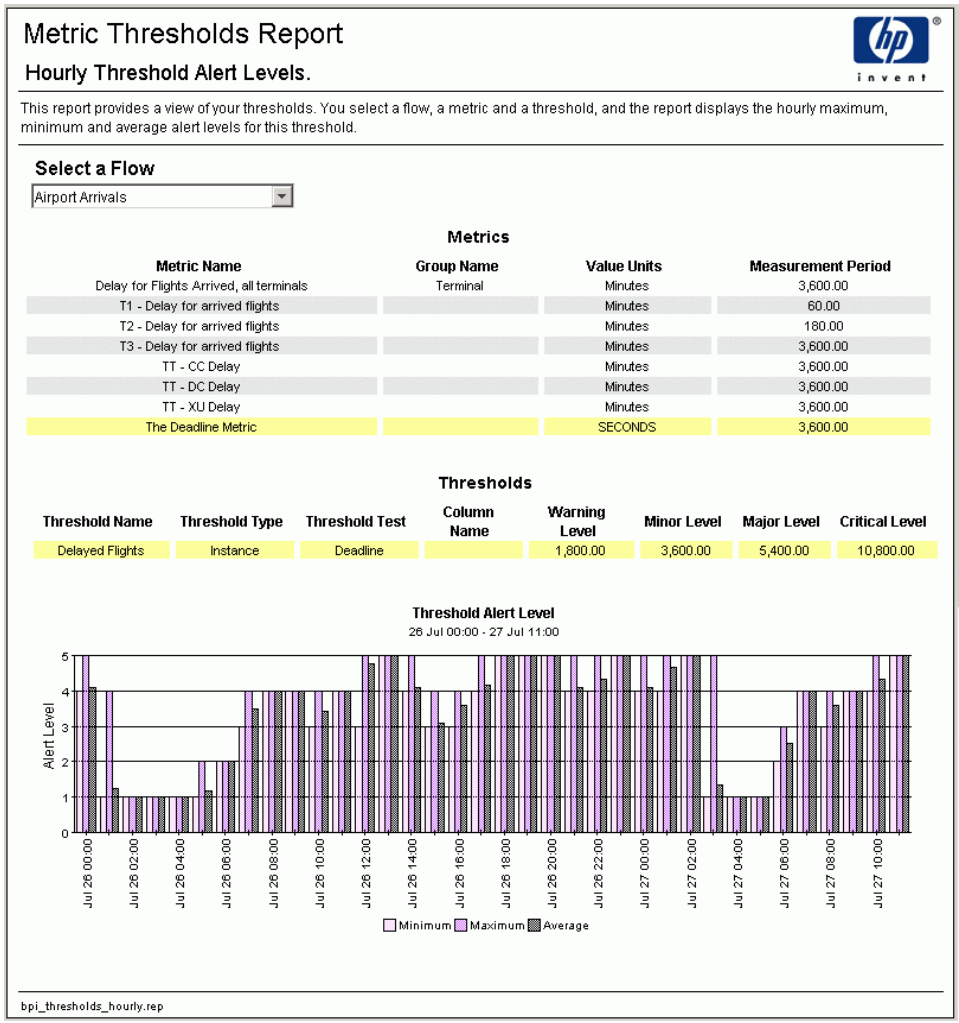
  For the alert levels graph.

  `S*` is either `SH`, `SD`, `SW`, or `SM` depending on whether the report is an hourly, daily, weekly or monthly report.

The datapipe that provides the data for these reports is the `BPIDP_THRESHOLDS` datapipe.

The trends used for these reports are named `BPIDP_*_THRESHOLDLEVELS.sum` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

Figure 17 Threshold Instance Summary Report



Metric Thresholds Report

Hourly Threshold Alert Levels.

This report provides a view of your thresholds. You select a flow, a metric and a threshold, and the report displays the hourly maximum, minimum and average alert levels for this threshold.

Select a Flow

Airport Arrivals

Metrics

Metric Name	Group Name	Value Units	Measurement Period
Delay for Flights Arrived, all terminals	Terminal	Minutes	3,600.00
T1 - Delay for arrived flights		Minutes	60.00
T2 - Delay for arrived flights		Minutes	180.00
T3 - Delay for arrived flights		Minutes	3,600.00
TT - CC Delay		Minutes	3,600.00
TT - DC Delay		Minutes	3,600.00
TT - XU Delay		Minutes	3,600.00
The Deadline Metric		SECONDS	3,600.00

Thresholds

Threshold Name	Threshold Type	Threshold Test	Column Name	Warning Level	Minor Level	Major Level	Critical Level
Delayed Flights	Instance	Deadline		1,800.00	3,600.00	5,400.00	10,800.00

Threshold Alert Level

26 Jul 00:00 - 27 Jul 11:00

bpi_thresholds_hourly.rep

Metric Alerts

The following metric alert reports are available:

- Metric alert summaries – Hourly/Daily/Weekly/Monthly

  Shows the overall number and distribution of alerts for a flow, metric and threshold.

- Metric Alerts for flow instances

  Refer to Flow Instance Listings Showing Alert Levels on page 78 and Flow Instance Alerts Summaries – Hourly/Daily/Weekly/Monthly on page 80 for details of these reports.

Let's now consider the overall metric alert summary reports.

Metric Alert Summaries – Hourly/Daily/Weekly/Monthly

Figure 18 on page 95 shows an example of the metric alert summary report. The report shown is an hourly summary. There are also reports for Daily, Weekly and Monthly summaries.

These summary reports enable you to see the overall number and distribution of alerts for a flow. You can then drill down to particular metrics and thresholds defined for this flow and see the alerts raised just for them.

The report file names are `bpi_metricAlerts_*.rep` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

The PI data tables used in these reports are as follows:

- `K_BPI_FLOWS`

  For the list of available flow names.

- `K_BPI_METRICS`

  For the list of metric names.

- `K_BPI_THRESHOLDS`

  For the list of threshold names.

- `K_BPI_FLOWALERTS/S*_BPI_FLOWALERTS`

  For the graph that shows the overall alert distribution for a given flow.

- `K_BPI_METRICALERTS/S*_BPI_METRICALERTS`

  For the graph that shows the overall alert distribution for a given metric.

- `K_BPI_THRESHOLDALERTS/S*_BPI_THRESHOLDALERTS`

  For the graph that shows the overall alert distribution for a given threshold. These tables are derived from the `R_BPI_METRICALERTDATA` table.

`S*` is either `SH`, `SD`, `SW`, or `SM` depending on whether the report is an hourly, daily, weekly or monthly report.

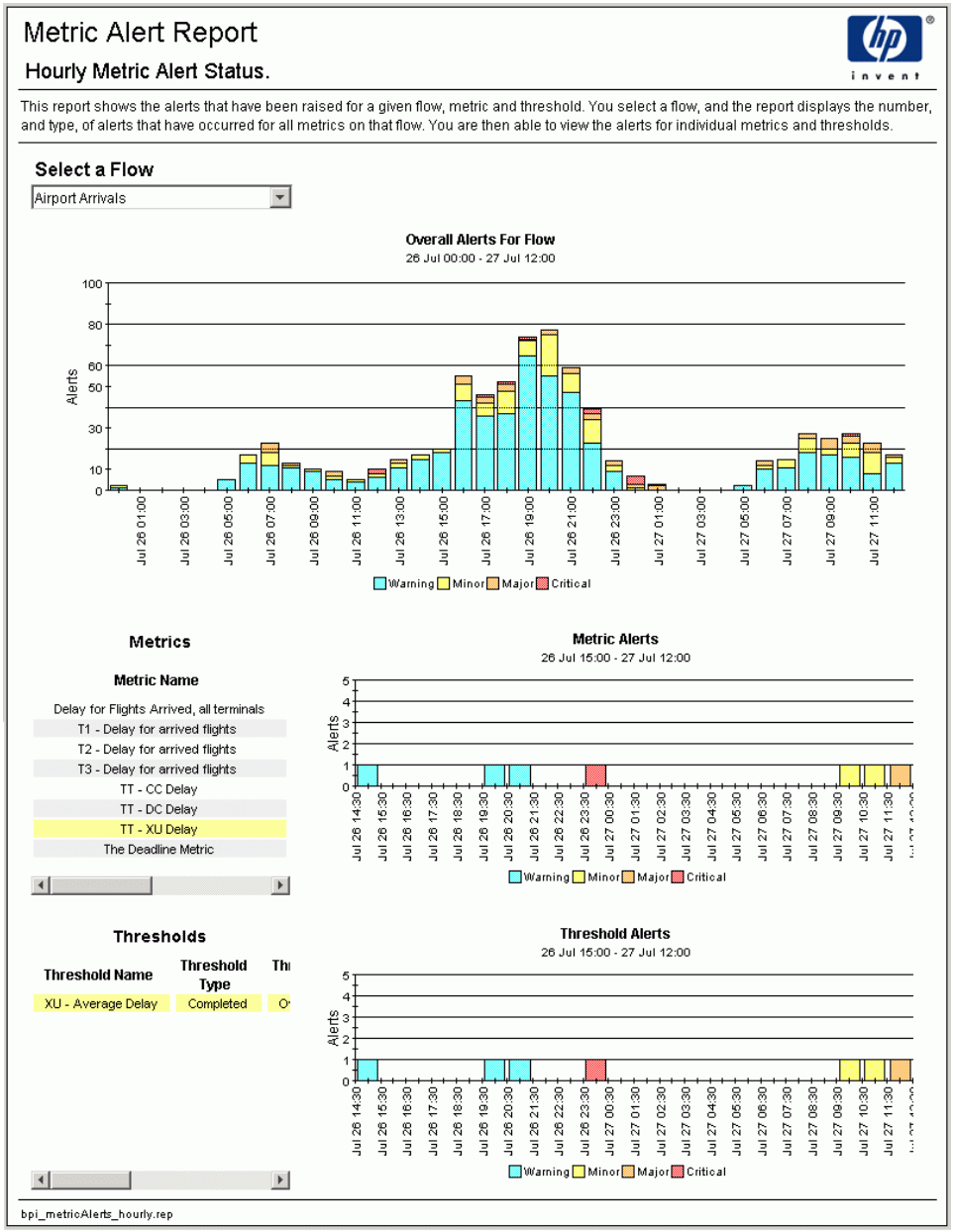
The datapipe that provides the data for these reports is the `BPIDP_METRICALERTS` datapipe.

The trends used for these reports are as follows:

- `BPIDP_*_FLOWALERTS.sum`
- `BPIDP_*_METRICALERTS.sum`
- `BPIDP_*_THRESHOLDALERTS.sum`

where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

Figure 18 Metric Alert Summary Report



Metric Statistics

The following metric statistics reports are available:

- Recent metric statistics details
- Metric statistics summaries – Hourly/Daily/Weekly/Monthly

Let's now consider some examples of these reports.

Recent Metric Statistics Details

Figure 19 on page 97 shows an example of the recent metric statistics report.

The report file name is `bpi_metricStats.rep`.

The PI data tables used in this report are as follows:

- `K_BPI_FLOWS/K_BPI_METRICS/K_BPI_THRESHOLDS`

  For the list of available flows, metrics and thresholds.

- `R_BPI_METRICSTATSDATA`

  For the list of metric group values.

- `RV_BPI_METRICSTATSPERPERIOD`

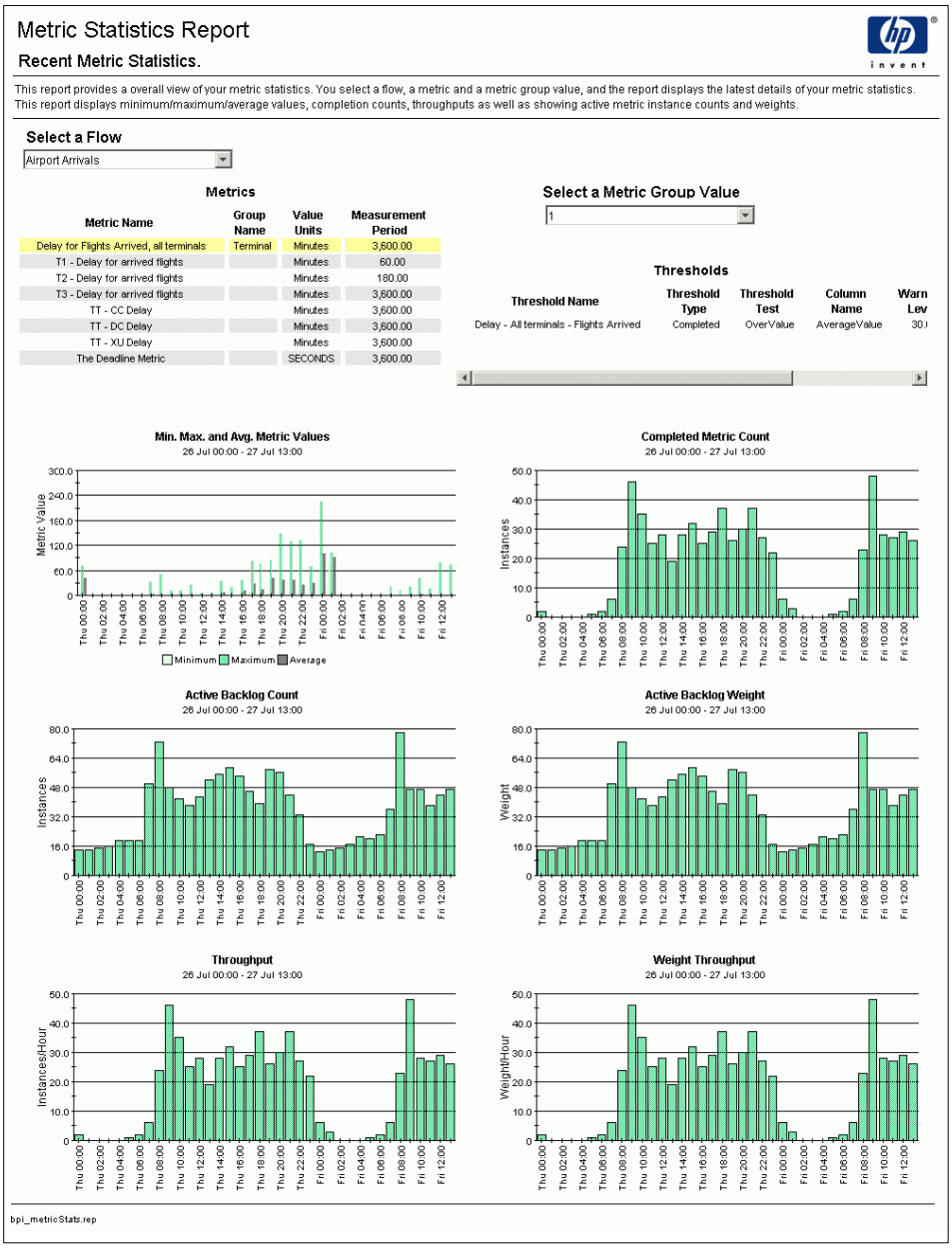
  For all the graphs.

  Because HPBPI metric statistics can be collected for periods of less than five minutes, reporting directly from the `R_BPI_METRICSTATSDATA` table can cause PI some problems. If all your HPBPI metric statistics collection intervals are greater then five minutes then you would not have a problem.

  The `RV_BPI_METRICSTATSPERPERIOD` view is a view onto the `R_BPI_METRICSTATSDATA` table that ensures that there is only a single metric statistics record for each PI time period.

The datapipe that provides the data for this report is the `BPIDP_METRICSTATS` datapipe.

Figure 19 Recent Metric Statistics Report



Metric Statistics Summaries – Hourly/Daily/Weekly/Monthly

Figure 20 on page 99 shows an example of the metric statistics summary report. The report shown is an hourly summary. There are also reports for Daily, Weekly and Monthly summaries.

The report file names are `bpi_metricStats_*.rep` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

The PI data tables used in these reports are as follows:

- `K_BPI_FLOWS/K_BPI_METRICS/K_BPI_THRESHOLDS`

  For the list of available flows, metrics and thresholds.

- `R_BPI_METRICSTATSDATA`

  For the list of metric group values.

- `K_BPI_METRICSTATS/S*_BPI_METRICSTATS`

  For all the graphs.

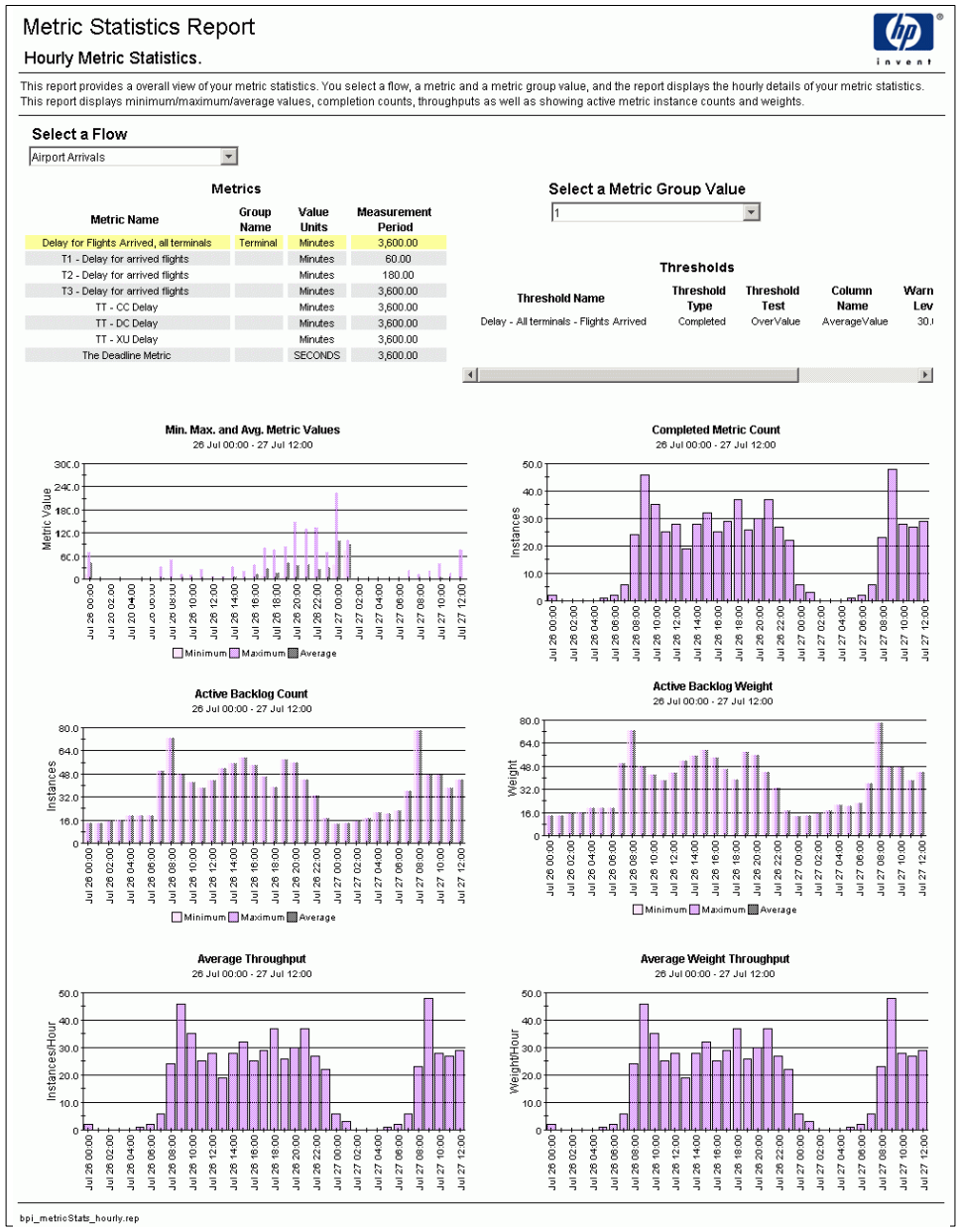
  These tables are derived from the `RV_BPI_METRICSTATSPERPERIOD` view.

  `S*` is either `SH`, `SD`, `SW`, or `SM` depending on whether the report is an hourly, daily, weekly or monthly report.

The datapipe that provides the data for these reports is the `BPIDP_METRICSTATS` datapipe.

The trends used for these reports are named `BPIDP_*_METRICSTATS.sum` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

Figure 20 Metric Statistics Summary Report



Metric Statistics Report

Hourly Metric Statistics.

This report provides a overall view of your metric statistics. You select a flow, a metric and a metric group value, and the report displays the hourly details of your metric statistics. This report displays minimum/maximum/average values, completion counts, throughputs as well as showing active metric instance counts and weights.

Select a Flow

Airport Arrivals

Metrics

Metric Name	Group Name	Value Units	Measurement Period
Delay for Flights Arrived, all terminals	Terminal	Minutes	3,600.00
T1 - Delay for arrived flights		Minutes	60.00
T2 - Delay for arrived flights		Minutes	180.00
T3 - Delay for arrived flights		Minutes	3,600.00
TT - CC Delay		Minutes	3,600.00
TT - DC Delay		Minutes	3,600.00
TT - XU Delay		Minutes	3,600.00
The Deadline Metric		SECONDS	3,600.00

Select a Metric Group Value

1

Thresholds

Threshold Name	Threshold Type	Threshold Test	Column Name	Warn Lev
Delay - All terminals - Flights Arrived	Completed	OverValue	AverageValue	30.(

Min. Max. and Avg. Metric Values

26 Jul 00:00 - 27 Jul 12:00

Completed Metric Count

26 Jul 00:00 - 27 Jul 12:00

Active Backlog Count

26 Jul 00:00 - 27 Jul 12:00

Active Backlog Weight

26 Jul 00:00 - 27 Jul 12:00

Average Throughput

26 Jul 00:00 - 27 Jul 12:00

Average Weight Throughput

26 Jul 00:00 - 27 Jul 12:00

bpi_metricStats_hourly.rep

Metric Values

The following metric values reports are available:

- Recent metric values list
- Metric values summaries – Hourly/Daily/Weekly/Monthly

Let's now consider some examples of these reports.

Recent Metric Values List

Figure 21 on page 101 shows an example of the recent metric values report.

The report file name is `bpi_metricValues.rep`.

The PI data tables used in this report are as follows:

- `K_BPI_FLOWS/K_BPI_METRICS/K_BPI_THRESHOLDS`

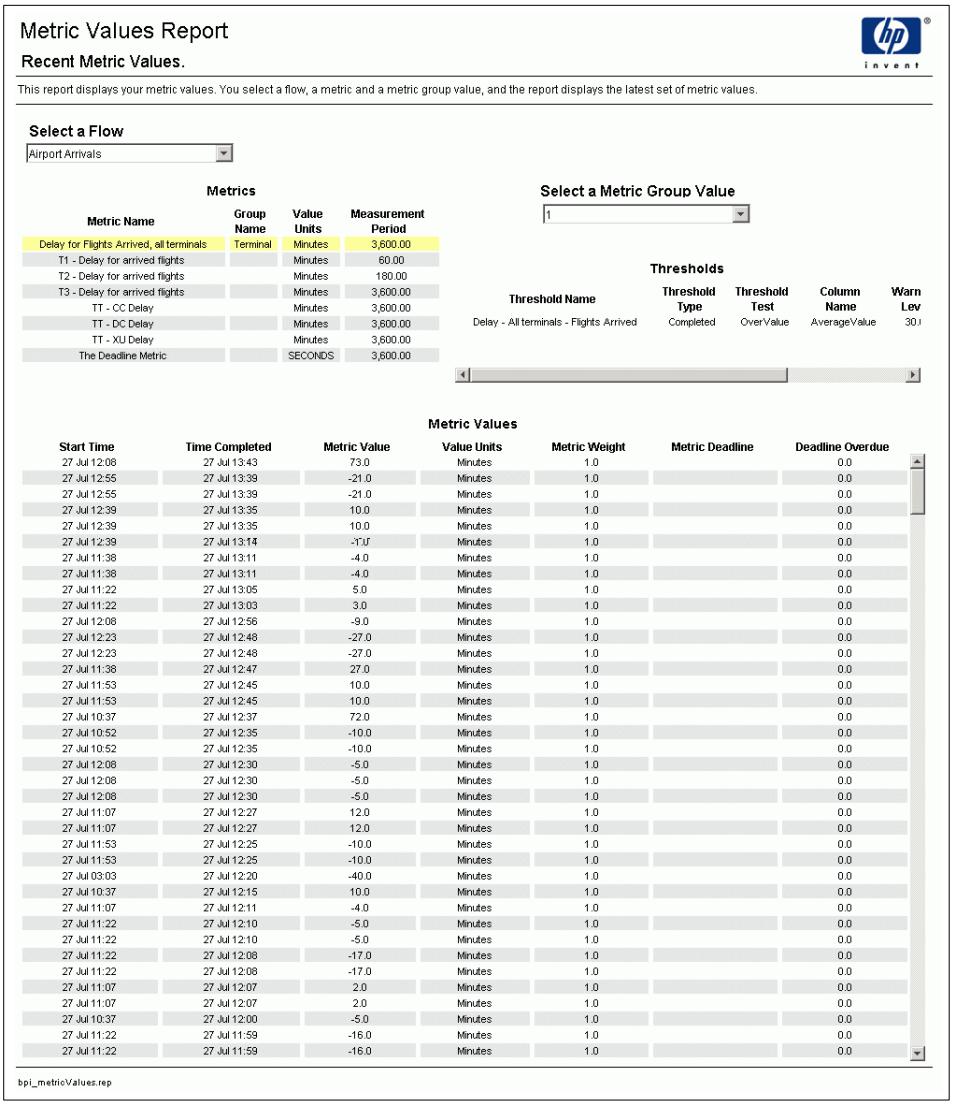
  For the list of available flows, metrics and thresholds.

- `R_BPI_METRICVALUEDATA`

  For the list of metric values.

The datapipe that provides the data for this report is the `BPIDP_METRICVALUES` datapipe.

Figure 21 Recent Metric Values Report



Metric Values Report

Recent Metric Values.

This report displays your metric values. You select a flow, a metric and a metric group value, and the report displays the latest set of metric values.

Select a Flow

Airport Arrivals

Metrics

Metric Name	Group Name	Value Units	Measurement Period
Delay for Flights Arrived, all terminals	Terminal	Minutes	3,600.00
T1 - Delay for arrived flights		Minutes	60.00
T2 - Delay for arrived flights		Minutes	180.00
T3 - Delay for arrived flights		Minutes	3,600.00
TT - CC Delay		Minutes	3,600.00
TT - DC Delay		Minutes	3,600.00
TT - XU Delay		Minutes	3,600.00
The Deadline Metric		SECONDS	3,600.00

Select a Metric Group Value

1

Thresholds

Threshold Name	Threshold Type	Threshold Test	Column Name	Warn Lev
Delay - All terminals - Flights Arrived	Completed	OverValue	AverageValue	30.(

Metric Values

Start Time	Time Completed	Metric Value	Value Units	Metric Weight	Metric Deadline	Deadline Overdue
27 Jul 12:08	27 Jul 13:43	73.0	Minutes	1.0		0.0
27 Jul 12:55	27 Jul 13:39	-21.0	Minutes	1.0		0.0
27 Jul 12:55	27 Jul 13:39	-21.0	Minutes	1.0		0.0
27 Jul 12:39	27 Jul 13:35	10.0	Minutes	1.0		0.0
27 Jul 12:39	27 Jul 13:35	10.0	Minutes	1.0		0.0
27 Jul 12:39	27 Jul 13:14	-1.0	Minutes	1.0		0.0
27 Jul 11:38	27 Jul 13:11	-4.0	Minutes	1.0		0.0
27 Jul 11:38	27 Jul 13:11	-4.0	Minutes	1.0		0.0
27 Jul 11:22	27 Jul 13:05	5.0	Minutes	1.0		0.0
27 Jul 11:22	27 Jul 13:03	3.0	Minutes	1.0		0.0
27 Jul 12:08	27 Jul 12:56	-9.0	Minutes	1.0		0.0
27 Jul 12:23	27 Jul 12:48	-27.0	Minutes	1.0		0.0
27 Jul 12:23	27 Jul 12:48	-27.0	Minutes	1.0		0.0
27 Jul 11:38	27 Jul 12:47	27.0	Minutes	1.0		0.0
27 Jul 11:53	27 Jul 12:45	10.0	Minutes	1.0		0.0
27 Jul 11:53	27 Jul 12:45	10.0	Minutes	1.0		0.0
27 Jul 10:37	27 Jul 12:37	72.0	Minutes	1.0		0.0
27 Jul 10:52	27 Jul 12:35	-10.0	Minutes	1.0		0.0
27 Jul 10:52	27 Jul 12:35	-10.0	Minutes	1.0		0.0
27 Jul 12:08	27 Jul 12:30	-5.0	Minutes	1.0		0.0
27 Jul 12:08	27 Jul 12:30	-5.0	Minutes	1.0		0.0
27 Jul 12:08	27 Jul 12:30	-5.0	Minutes	1.0		0.0
27 Jul 11:07	27 Jul 12:27	12.0	Minutes	1.0		0.0
27 Jul 11:07	27 Jul 12:27	12.0	Minutes	1.0		0.0
27 Jul 11:53	27 Jul 12:25	-10.0	Minutes	1.0		0.0
27 Jul 11:53	27 Jul 12:25	-10.0	Minutes	1.0		0.0
27 Jul 03:03	27 Jul 12:20	-40.0	Minutes	1.0		0.0
27 Jul 10:37	27 Jul 12:15	10.0	Minutes	1.0		0.0
27 Jul 11:07	27 Jul 12:11	-4.0	Minutes	1.0		0.0
27 Jul 11:22	27 Jul 12:10	-5.0	Minutes	1.0		0.0
27 Jul 11:22	27 Jul 12:10	-5.0	Minutes	1.0		0.0
27 Jul 11:22	27 Jul 12:08	-17.0	Minutes	1.0		0.0
27 Jul 11:22	27 Jul 12:08	-17.0	Minutes	1.0		0.0
27 Jul 11:07	27 Jul 12:07	2.0	Minutes	1.0		0.0
27 Jul 11:07	27 Jul 12:07	2.0	Minutes	1.0		0.0
27 Jul 10:37	27 Jul 12:00	-5.0	Minutes	1.0		0.0
27 Jul 11:22	27 Jul 11:59	-16.0	Minutes	1.0		0.0
27 Jul 11:22	27 Jul 11:59	-16.0	Minutes	1.0		0.0

bpi_metricValues.rep

Metric Values Summaries – Hourly/Daily/Weekly/Monthly

Figure 22 on page 103 shows an example of the metric values summary report. The report shown is an hourly summary. There are also reports for Daily, Weekly and Monthly summaries.

The report file names are `bpi_metricValues_*.rep` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

The PI data tables used in these reports are as follows:

- `K_BPI_FLOWS/K_BPI_METRICS/K_BPI_THRESHOLDS`

  For the list of available flows, metrics and thresholds.

- `R_BPI_METRICVALUEDATA`

  For the list of metric group values.

- `K_BPI_METRICVALUES/S*_BPI_METRICVALUES`

  For all the graphs.

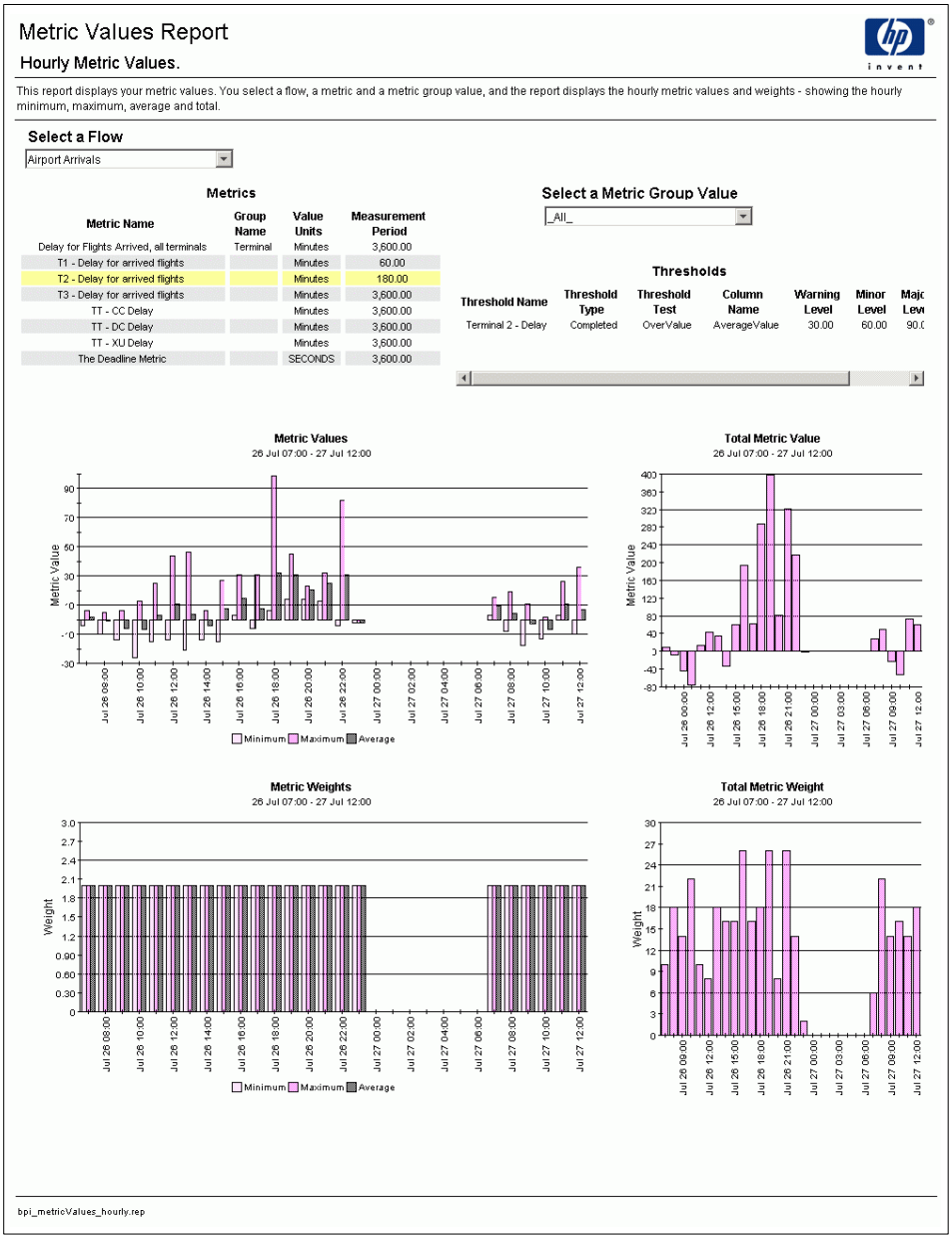
  These tables are derived from the `R_BPI_METRICVALUEDATA` table.

  `S*` is either `SH`, `SD`, `SW`, or `SM` depending on whether the report is an hourly, daily, weekly or monthly report.

The datapipe that provides the data for these reports is the `BPIDP_METRICVALUES` datapipe.

The trends used for these reports are named `BPIDP_*_METRICVALUES.sum` where `*` is either `hourly`, `daily`, `weekly` or `monthly`.

Figure 22 Metric Values Summary Report



5 Producing Custom Reports

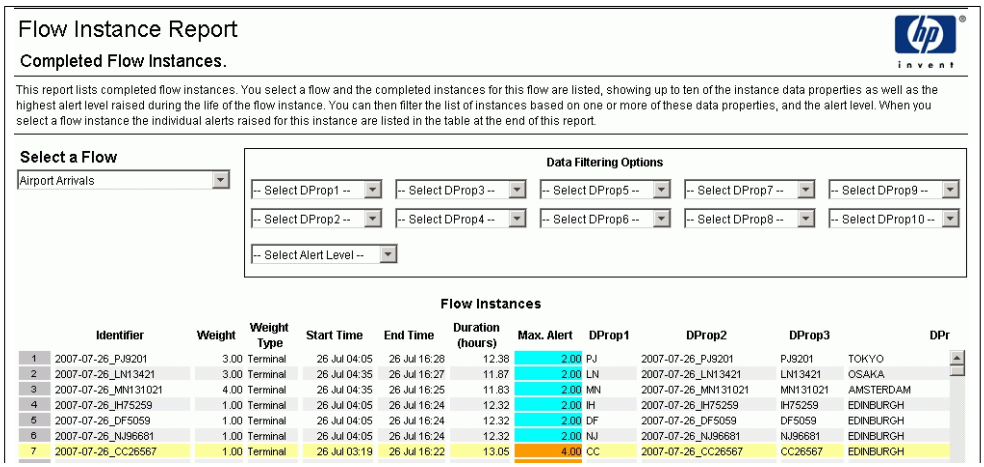
The reports that come with the HPBPI Report Pack are generic reports that are able to display flow, node and metric information for any HPBPI flow.

This chapter looks at some examples of how to produce custom reports for your specific flows.

Flow Specific Reporting

The HPBPI Report Pack provides a report that displays the latest set of flow instances along with the highest alert level they achieved. An example of this report is shown in Figure 23 and the report is described in detail in the section Flow Instance Listings Showing Alert Levels on page 78.

Figure 23 Standard Flow Instance Report



The flow instance listing report shown in Figure 23 is a generic report that lets you select a flow and then, optionally, filter the instances that are displayed by selecting a combination of up to 10 business data attribute values.

The data filtering drop-down boxes offer the business data attributes with the text `-- Select DProp`*`N`*` --`, and the business data for each flow instance is displayed in columns that are headed with the text `DProp`*`N`*, where *N* is the number `1` to `10`.

The values within each of these business data attributes varies from flow to flow depending on what the flow is monitoring. For example, if you have an order flow, that tracks customer orders, the business data attributes may contain details such as the customer name, address and email. However, for a flow monitoring airport departures, the business data attributes may contain data such as the flight carrier, the airport terminal and the flight destination.

Once you know that a report is only displaying the flow instances for a single HPBPI flow, you know exactly what data is in each of the business data attributes, and so you can use more meaningful names within the report.

You can use the standard flow instance listing report (as shown in Figure 23 on page 106) to select your flow and then look at the data within each of the 10 business data attributes. This allows you to determine the real names for these attributes.

Let's look at an example of how to produce a report for a specific flow.

Flight Departures

In this example, the report is for the flow called: `Airport Departures`.

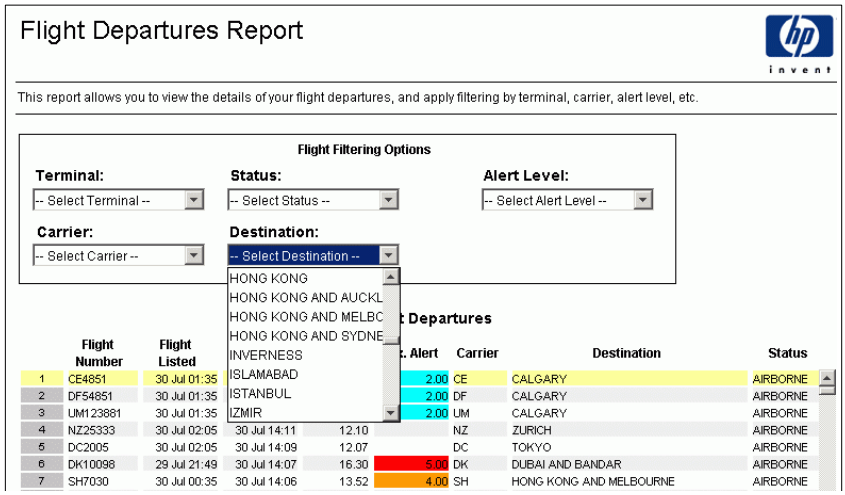
You decide that the business data attributes you wish to show in your report are as follows:

- `dprop1` which is the `Flight Carrier`.
- `dprop4` which is the `Flight Destination`.
- `dprop5` which is the `Flight Status`.
- `dprop6` which is the `Airport Terminal`.

You can then define a database view within PI that selects all the data from the flow instance table (`R_BPI_FLOWINSTANCEDATA`) and the metric alert table (`R_BPI_METRICALERTDATA`) where the flow name is `Airport Departures`.

You can then define a custom report that reports on this view and presents the data as shown in Figure 24.

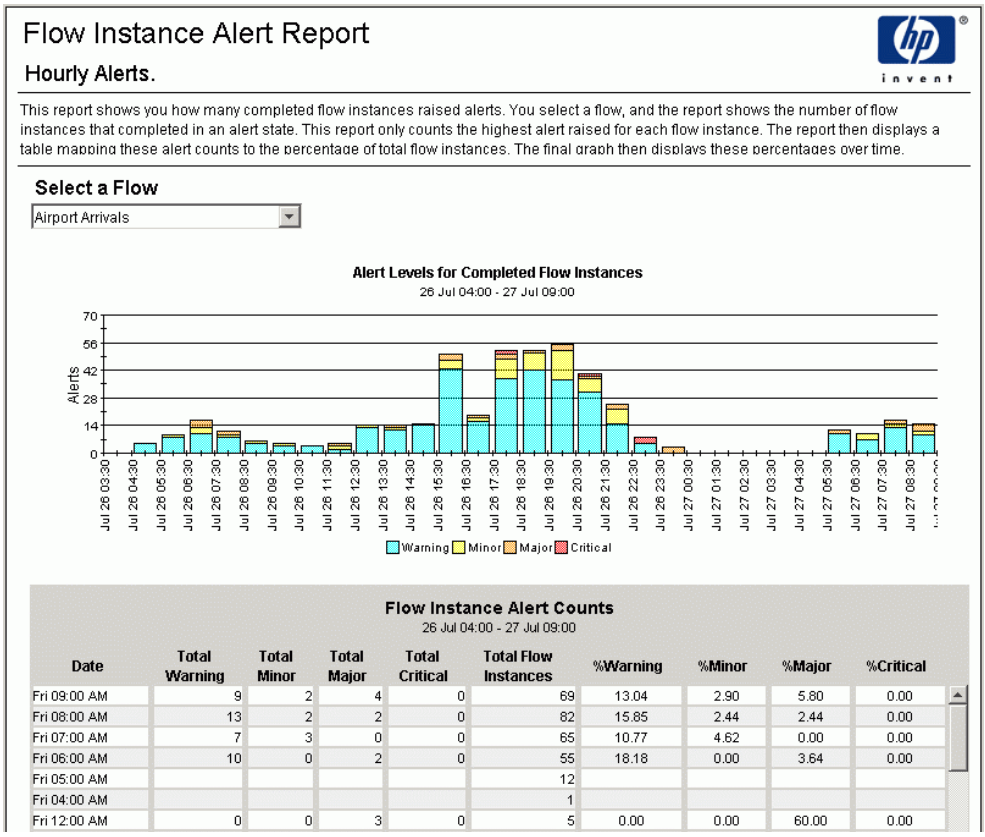
Figure 24 Flight Departures Custom Report



Reporting Grouped by Business Data

The HPBPI Report Pack provides an hourly summary report that displays the number and distribution of alerts thrown by flow instances. An example of this report is shown in Figure 25 and the report is described in detail in the section Flow Instance Alerts Summaries – Hourly/Daily/Weekly/Monthly on page 80.

Figure 25 Standard Flow Instance Alert Hourly Report



Date	Total Warning	Total Minor	Total Major	Total Critical	Total Flow Instances	%Warning	%Minor	%Major	%Critical
Fri 09:00 AM	9	2	4	0	69	13.04	2.90	5.80	0.00
Fri 08:00 AM	13	2	2	0	82	15.85	2.44	2.44	0.00
Fri 07:00 AM	7	3	0	0	65	10.77	4.62	0.00	0.00
Fri 06:00 AM	10	0	2	0	55	18.18	0.00	3.64	0.00
Fri 05:00 AM					12				
Fri 04:00 AM					1				
Fri 12:00 AM	0	0	3	0	5	0.00	0.00	60.00	0.00

The flow instance alert report shown in Figure 25 is a generic report that lets you select the flow. The report then displays the alert distribution across all flights for that hour.

Suppose you want to be able to report this alert data grouped by business data attributes?

Flight Departure Alerts by Terminal/Carrier

In this example, the report is for a flow called: `Airport Departures`.

You would like to produce a report that allows you to show hourly flow instance alert data grouped by airport terminal, and then grouped by flight carrier.

Using the standard flow reports you determine that the business data attributes you require are as follows:

- `dprop6` which is the `Airport Terminal`.
- `dprop1` which is the `Flight Carrier`.

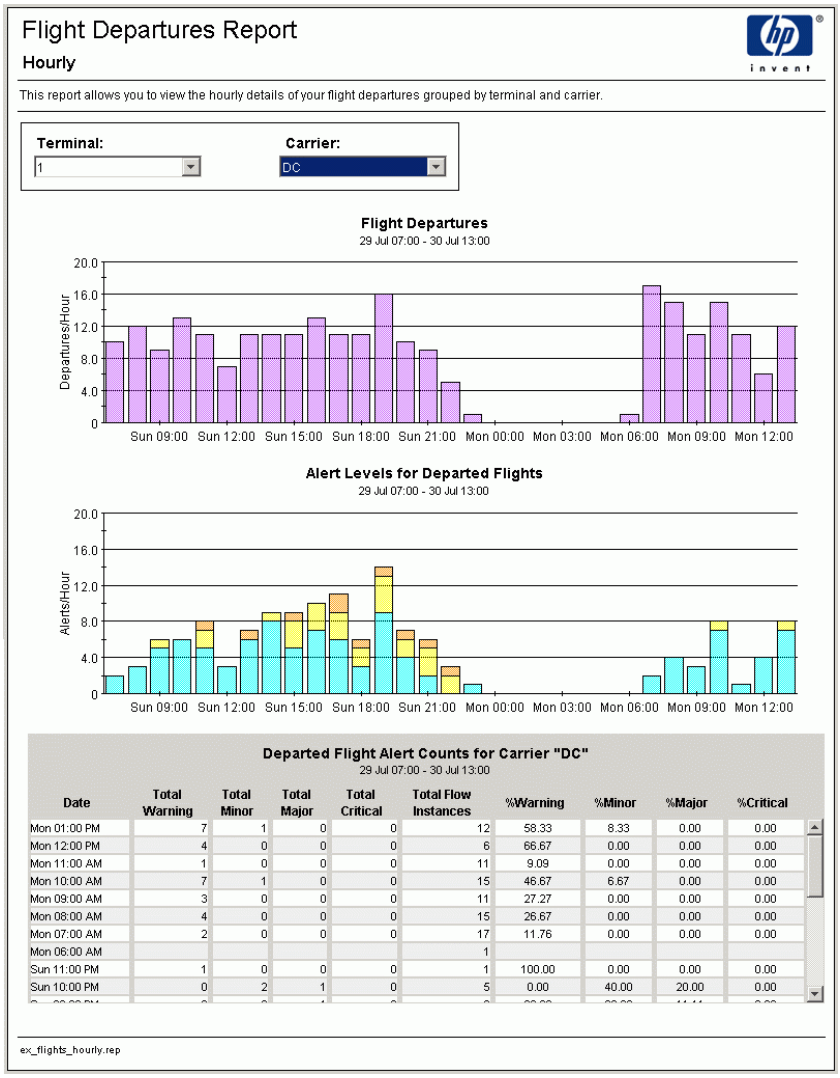
You can then define a database view within PI that selects all the data from the flow instance table (`R_BPI_FLOWINSTANCEDATA`) and the metric alert table (`R_BPI_METRICALERTDATA`) where the flow name is `Airport Departures`. Let's call this view `RV_BPI_FLIGHTDEPARTURES`.

Now you can set up a trend_sum that trends this `RV_BPI_FLIGHTDEPARTURES` view, grouping the data by `dprop6` (the airport terminal) and `dprop1` (the fight carrier). For example:

```
source table:RV_BPI_FLIGHTDEPARTURES
destination table:SH_BPI_FLIGHTS

column:!countWarning=alertLevel:vct[2]
column:!countMinor=alertLevel:vct[3]
column:!countMajor=alertLevel:vct[4]
column:!countCritical=alertLevel:vct[5]

column:!totalDuration=duration:tot
column:!maxDuration=duration:max
column:!minDuration=duration:min
column:!avgDuration=duration:avg

by variable:dProp6
by variable:dProp1
by variable:hour
```

You can now define a custom report as shown in Figure 26 on page 110. This report displays the alert statistics for the combination of airport terminal and flight carrier selected by the user.

Figure 26 Flight Departures by Terminal and by Carrier



Date	Total Warning	Total Minor	Total Major	Total Critical	Total Flow Instances	%Warning	%Minor	%Major	%Critical
Mon 01:00 PM	7	1	0	0	12	58.33	8.33	0.00	0.00
Mon 12:00 PM	4	0	0	0	6	66.67	0.00	0.00	0.00
Mon 11:00 AM	1	0	0	0	11	9.09	0.00	0.00	0.00
Mon 10:00 AM	7	1	0	0	15	46.67	6.67	0.00	0.00
Mon 09:00 AM	3	0	0	0	11	27.27	0.00	0.00	0.00
Mon 08:00 AM	4	0	0	0	15	26.67	0.00	0.00	0.00
Mon 07:00 AM	2	0	0	0	17	11.76	0.00	0.00	0.00
Mon 06:00 AM					1				
Sun 11:00 PM	1	0	0	0	1	100.00	0.00	0.00	0.00
Sun 10:00 PM	0	2	1	0	5	0.00	40.00	20.00	0.00

Index

I

M

N

O

P

R

S

T

U