Operational Analyst

Release 6.1

Data API Utility

585-248-603
Issue 1.0
August 2003

Notice

While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, Avaya Inc. can assume no liability for any errors. Changes and corrections to the information in this document may be incorporated in future releases.

Preventing toll fraud

"Toll fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, anyone who is not a corporate employee, agent, subcontractor, or person working on your company's behalf). Be aware that there may be a risk of toll fraud associated with your system and that, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya fraud intervention

If you suspect that you are being victimized by toll fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Web site:

http://www.avaya.com

Select **Support**, then select **Escalation Lists**. This Web site includes telephone numbers for escalation within the United States. For escalation telephone numbers outside the United States, select **Global Escalation List**.

Providing telecommunications security

Telecommunications security (of voice, data, and video communications) is the prevention of any type of intrusion to (that is, either unauthorized or malicious access to or use of) your company's telecommunications equipment by some party.

Your company's "telecommunications equipment" includes both this Avaya product and any other voice/data/video equipment that could be accessed via this Avaya product (that is, "networked equipment").

An "outside party" is anyone who is not a corporate employee, agent, subcontractor, or person working on your company's behalf. Whereas, a "malicious party" is anyone (including someone who may be otherwise authorized) who accesses your telecommunications equipment with either malicious or mischievous intent.

Such intrusions may be either to/through synchronous (time-multiplexed and/or circuit-based) or asynchronous (character-, message-, or packet-based) equipment or interfaces for reasons of:

- Use (of capabilities special to the accessed equipment)
- Theft (such as, of intellectual property, financial assets, or toll-facility access)
- Eavesdropping (privacy invasions to humans)
- Mischief (troubling, but apparently innocuous, tampering)
- Harm (such as harmful tampering, data loss or alteration, regardless of motive or intent)

Be aware that there may be a risk of unauthorized intrusions associated with your system and/or its networked equipment. Also realize that, if such an intrusion should occur, it could result in a variety of losses to your company (including, but not limited to, human and data privacy, intellectual property, material assets, financial resources, labor costs, and legal costs).

Your responsibility for your company's telecommunications security

The final responsibility for securing both this system and its networked equipment rests with you, an Avaya customer's system administrator, your telecommunications peers, and your managers. Base the fulfillment of your responsibility on acquired knowledge and resources from a variety of sources, including, but not limited to:

- Installation documents
- System administration documents
- Security documents
- Hardware-/software-based security tools
- Shared information between you and your peers
- Telecommunications security experts

To prevent intrusions to your telecommunications equipment, you and your peers should carefully program and configure:

- Your Avaya-provided telecommunications systems and their interfaces
- Your Avaya-provided software applications, as well as their underlying hardware/software platforms and interfaces
- Any other equipment networked to your Avaya products.

Warranty

Avaya Inc. provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available through the following Web site:

http://www.avaya.com/support

Link disclaimer

Avaya Inc. is not responsible for the contents or reliability of any linked Web sites and does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all of the time and we have no control over the availability of the linked pages.

Trademarks

Avaya is a trademark of Avaya Inc.

All trademarks identified by the ® or ™ are registered trademarks or trademarks, respectively, of Avaya Inc. All other trademarks are the property of their respective owners.

Ordering information: Avaya Publications Center

Voice: +1-207-866-6701
1-800-457-1764 (Toll-free, U.S. and Canada only)

Fax: +1-207-626-7269
1-800-457-1764 (Toll-free, U.S. and Canada only)

Write: Globalware Solutions
200 Ward Hill Avenue
Haverhill, MA 01835 USA
Attention: Avaya Account Manager

Web: http://www.avayadocs.com

E-mail: totalware@gwsmail.com

Order: Document No. 585-248-603, Issue 1.0
August 2003

Avaya support

Avaya provides a telephone number for you to use to report problems or to ask questions about your contact center. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site:

http://www.avaya.com

Select **Support**, then select **Escalation Lists**. This Web site includes telephone numbers for escalation within the United States. For escalation telephone numbers outside the United States, select **Global Escalation List**.

Comments

To comment on this document, send e-mail to crminfodev@avaya.com.

Acknowledgment

This document was written by the CRM Information Development group.

Avaya Operational Analyst
Release 6.1
Data API Utility

Contents

Preface 5

Purpose of this document 6
Intended users of this document. 6
Overview of this document 6
Safety labels 7
Reasons for reissuing this document 7
Availability of this document. 7
Related documentation 8

Data API Utility 9

Installation 9
Security 10
User Authentication 10
User authentication troubleshooting 10
Data model 11

Using the Data API Utility 13

Parameters 14
Parameters explained 15
Parameter substitution 18
Usage examples 19
Getting the usage instructions 19
Saving the userid and password 20
Getting historical data 20
Getting real-time updates 21
Specifying data output and logs location 21

Data API Utility query limits 23

Consolidated data 24
Real-time SQL limitations 25
Example 25
Data Set Characteristics 26
OA historical data 27
Examples: 27

Data Export API performance information 29

Index 31

Preface

Avaya Operational Analyst Release 6.1 Data API Utility is a command line program for exporting real-time or historical data from the Operational Analyst (OA) reporting framework. The Utility returns data for SQL SELECT statements as either XML or delimiter-separated text. The Utility is a java program that calls the Data Export API. The Data Export API is a component of the Report Framework.

Note:

Contact your Avaya Professional Services representative for information about the Data Export API.

This section includes the following topics:

- Purpose of this document on page 6
- Intended users of this document on page 6
- Overview of this document on page 6
- Safety labels on page 7
- Reasons for reissuing this document on page 7
- Availability of this document on page 7
- Related documentation on page 8

Purpose of this document

The purpose of this document is to describe the operation and operating environment of the OA R6.1 Data API Utility.

Intended users of this document

This document is written for:

- Contact Center system administrators.
- Avaya Professional Services representatives.
- Contact Center managers or supervisors who have a valid user ID and password to logon to the Reporting Framework host.

Overview of this document

This document includes the following topics:

- Data API Utility on page 9 - Introduces the Data API Utility and discusses installation and security requirements.
- Using the Data API Utility on page 13 - Demonstrates how to use the Data API Utility, including parameter definitions and substitution rules.
- Data API Utility query limits on page 23 - Defines and shows examples of the query limitations of the Data API Utility.
- Data Export API performance information on page 29 - Provides examples of methods you can use to maximize the performance of the Data API Utility.
- Index on page 31 – Lists topics covered in this document, giving page numbers where each topic is mentioned.

Safety labels

If you see any of the following safety labels in this document, take careful note of the information presented.

⚠ **CAUTION:**
Caution statements call attention to situations that can result in harm to software, loss of data, or an interruption in service.

⚠ **WARNING:**
Warning statements call attention to situations that can result in harm to hardware or equipment.

⚠ **DANGER:**
Danger statements call attention to situations that can result in harm to personnel.

⚠ **SECURITY ALERT:**
Security alert statements call attention to situations that can increase the potential for unauthorized use of a telecommunications system.

Reasons for reissuing this document

This is the first issue of this document.

Availability of this document

Copies of this document are available from one or both of the following sources. Although there is no charge for documents obtained through the Avaya Web site, documents obtained through the Avaya Publications Center must be purchased.

- The Avaya online support Web site, http://www.avayadocs.com.
- The Avaya Publications Center:

Voice: +1-207-866-6701
+1-800-457-1764 (Toll-free, U.S. and Canada only)

Fax: +1-207-626-7269
+1-800-457-1764 (Toll-free, U.S. and Canada only)

Write: Globalware Solutions
200 Ward Hill Avenue
Haverhill, MA 01835 USA
Attention: Avaya Account Manager

E-mail: totalware@gwsmail.com

Related documentation

You might find the following documentation useful.

Avaya documentation:

Avaya documentation is available through the Avaya online support Web site, http://www.avayadocs.com.

- *Avaya Operational Analyst Release 6.1 Installation Planning and Prerequisites,* Issue 1.0, August 2003, Avaya document #585-248-118.
- *Avaya Operational Analyst Release 6.1 Installation and Configuration,* Issue 1.0, August 2003, Avaya document #585-248-119.
- *Avaya Operational Analyst Release 6.1 Maintenance and Troubleshooting,* Issue 1.0, August 2003, Avaya document #585-248-120.

Other documentation:

- User guides or troubleshooting information supplied by your database manufacturer.

Data API Utility

The Operational Analyst Release 6.1 (OA R6.1) Data API Utility (Utility) is a command line program for exporting real-time or historical data from the OA Reporting Framework. The Utility returns data for SQL SELECT statements as either XML or delimiter-separated text. The Utility is a java program that calls the Data Export API, a component of the Report Framework.

Note:

Contact your Avaya Professional Services representative for information about the Data Export API.

This section includes the following topics:

- Installation on page 9
- Security on page 10
- Data model on page 11

Installation

The Utility is installed using the OA R6.1 installation program. It can be installed on any of the OA hosts or by itself, on Windows, Solaris or AIX. Once installed, the Utility is available to any user that is able to access the OA applications (members of the OA administration group). All the required software components are installed automatically and no specific configuration is needed to run the Utility.

Note:

Refer to *Avaya Operational Analyst Release 6.1 Installation and Configuration* for full installation information.

Security

The Utility provides security for your data.

This section includes the following topics:

- User Authentication on page 10
- User authentication troubleshooting on page 10

User Authentication

The userid and password are recorded (password is encrypted) on the local drive when you run the Utility with the **-s** option. This option only needs to be used the first time you run the Utility and when the password changes.

The userid and password are authenticated against the security service of the Report Framework. This is done transparently every time that the Utility is used.

User authentication troubleshooting

These are steps you should follow if the authentication fails:

1. Verify that the userid and password are recorded where the Utility is installed. To do that, re-run the Utility with the **-s** option described in Parameters explained on page 15. Try again. If authentication still fails, go to step 2.
2. Try to log on to the Report Framework host using the userid and password that you have recorded. On a Windows machine, you can map a drive to do this. On a Solaris or AIX machine, you can create a telnet or FTP session to that host to do this. If you can't login to the Report Framework host, ask the administrator to fix the problem with the userid/password and then try again. If you can log on to the Report Framework host, go to step 3.
3. Look at the Utility logs. They may give you information about what may be going wrong. The log file is located in **data/log/DataExportUtility** where OA is installed. If this doesn't help, go to step 4.
4. Have an OA administrator verify the Report Framework logs. This is documented in *Avaya Operational Analyst Release 6.1 Maintenance and Troubleshooting*. If the administrator can't diagnose the problem, go to step 5.
5. Contact Avaya support.

Data model

The data model is available on the product CD. Please refer to **`PABASE/doc/Data_Models/index.html`** on the OA host for details of real-time and historical data models. **`PABASE`** is where the OA software is installed, typically **`/export/home/biadmin`** on Solaris platforms, **`/home/biadmin`** on AIX platforms, and **`C:\Program Files\Avaya\BI`** on Windows platforms.

Note:

The data models are also available on the OA R6.1 product CD at **`doc/Data_Models/index.html`**

Using the Data API Utility

The interface of the Utility is a command line. The Utility is presented as a shell or batch file that calls the appropriate Java executable with any arguments necessary. The following topics document the use of the Utility.

This section includes the following topics:

- Parameters on page 14
- Parameters explained on page 15
- Parameter substitution on page 18
- Usage examples on page 19

Parameters

Note:

Square brackets indicate optional arguments, round brackets group options, and pipes indicate that only one of the options in that group may be selected.

```
DataExportUtility
        [-n | --noecho]
        [(-l logfile | --log logfile)]
        (-s username | --store username)

DataExportUtility
        [-n | --noecho]
        [(-l logfile | --log logfile)]
        (-q "querystring" | --query "querystring")
        (((-a "timewindow" | --active "timewindow") |
          (-u period "timewindow" [updates] |
            --update period "timewindow" [updates]) |
          [-e | --extra])
          (-h | --historical))
        ((-x schemafile | --xml schemafile) |
          (-d ["delimiter"] | --delimited "delimiter"))
        (-p port | --port port)
        (-o resultfile | --output resultfile)
        server
```

`server` indicates which OA reports server to connect to when attempting to export data.

The Report Framework must be up and running on the target host because the Data Export API depends on the Report Framework to run properly. This server must also have a historical database or real-time subsystem configured.

Note:

See *Avaya Operational Analyst Release 6.1 Installation Planning and Prerequisites* for information about configuring the historical database, and *Avaya Operational Analyst Release 6.1 Installation and Configuration* for details of configuring a real-time subsystem.

In order to connect, a username and password must be stored by executing the Utility with the **`-s`** option. Only one username and password are stored by the Utility.

Parameters explained

Parameter	Explanation
`-n | -noecho`	No output other than the data will be sent to standard out. This feature is particularly useful for users that will capture the command-line output when running the Utility from another program.
`-s username | --store username`	prompts for and stores a password for the given **`username`**. The username and password must represent a valid user according to the OA authentication server. The username and password given replace any previously stored values. Refer to Security on page 10 for details.
`-l logfile | --log logfile`	specifies the file to log any error messages to.
`-q querystring | --query querystring`	specifies the SQL select statement to use. Only queries supported by the historical database and RDS are allowed. See Data API Utility query limits on page 23 for details. This statement is subject to parameter substitution as defined in Parameter substitution on page 18.
`-a timewindow | --active timewindow`	specifies that the query should be run against the RDS to retrieve real-time data. **`timewindow`** indicates which of the various time windows supported by the RDS are to be used. The timewindow may only be the id of one of **B**, **D1**, **D2**, **D3** or **D4**, which represent the windows configured on the RDS. See Data Set Characteristics on page 26 for information about time windows.

Parameter	Explanation
`-u period timewindow [updates] | --update period timewindow [updates]`	specifies that the query should be run against the RDS to retrieve several updates of real-time data. **`period`** indicates how many seconds the Utility should wait between attempts to retrieve new data, and tells the RDS what spacing the updates to data should be at minimum. The period may only be one of **5**, **10**, **15**, **30**, **60**, **300**, **600**, **900**. **`timewindow`** indicates which of the various time windows supported by the RDS are to be used. The timewindow may only be the one of **B**, **D1**, **D2**, **D3** or **D4**, which represent the windows configured on the RDS. **`updates`** indicates how many updates the Utility should perform before exiting. Leaving `updates` unspecified, or setting this to a negative number indicates that the Utility should run forever.
`-e | --extra`	indicates whether the time interval calculation columns are included in the real-time result. These columns are: ● **B** which represents the start time of the base interval, and ● **D1**, **D2**, **D3** or **D4** which represent the start time of the **timewindow** specified by the **-a** or **-u** option. The extra columns can be used to calculate interval time.
`-h | --historical`	specifies that the query should be run against the historical database.
`-x schemafile | --xml schemafile`	specifies that the output should be XML formatted. **`schemafile`** specifies the file that the XML schema of the results will be written to.
`-d [delimiter] | -delimited [delimiter]`	specifies that the output should be delimiter-separated. If **`delimiter`** is not specified, it is assumed to be a comma (,).

Parameter	Explanation
`-p port | --port port`	specifies the port that the server is running on. **`port`** defaults to **80** if the server is using HTTP, and **443** if the server is using HTTPS.
`-o resultfile | --output resultfile`	specifies the file that the results will be written to. If **`resultfile`** is "**-**", or this option is not specified, then output defaults to standard out. **Note:** The filename is subject to parameter substitution as documented in Parameter substitution on page 18. The files are opened to overwrite any previous contents. If updates of active data are requested (through the **`-u`** or **`--update`** option), and the filename given does not contain substitutions, then the result file contains only the results of the last update.

Note:

If any option is specified twice, or two incompatible options are specified, then the last option is used, overwriting the first option.

Parameter substitution

The SQL query and the output filename may contain special characters that may be substituted as follows:

Character	Meaning
%d	yesterday's date (e.g., 20000601 if today is June 2, 2000)
%D	today's date (e.g., 20000602 if today is June 2, 2000)
%h	the previous hour (e.g., 17 if the current time is between 6:00 and 6:59 p.m.)
%H	the current hour (e.g., 18 if the current time is between 6:00 and 6:59 p.m.)
%i	the start time for the previous half hour interval (e.g., 1730 if the current time is between 6:00 and 6:29 p.m.)
%I	the start time for the current half hour interval (e.g., 1800 if the current time is between 6:00 and 6:29 p.m.)
%r	the row date for the previous half hour interval (e.g., 200006021730 if the time is between 6:00 and 6:29 p.m. on June 2, 2000)
%R	the row date for the current half hour interval (e.g., 200006021800 if the time is between 6:00 and 6:29 p.m. on June 2, 2000)
%%	a literal %

In an SQL select statement, the date and time are converted into UTC. For example **%D, %H** at 9:30 am, September 2, 2002 Australian Eastern Standard Time is converted to **20020901, 23**.

Usage examples

The following examples can be run from Windows, Solaris or AIX platforms. They assume that:

- You have a valid OA userid on the host where the Utility is installed and where the Report Framework is installed.
- The OA user profile is properly setup as documented in *Avaya Operational Analyst Release 6.1 Installation and Configuration*.

Note:

Quotes are different on Unix platforms. You need to use the single quote (`'`) on Solaris and AIX.

This section includes the following topics:

- Getting the usage instructions on page 19
- Saving the userid and password on page 20
- Getting historical data on page 20
- Getting real-time updates on page 21
- Specifying data output and logs location on page 21

Getting the usage instructions

Simply type the script name, with no parameter.

```
C:\>DataExportUtility
Starting Data API Utility...

Use:
No mode selected.

DataExportUtility
        [-n                     | --noecho]
        [(-l logfile            | --log logfile)]
        (-s username            | --store username)

DataExportUtility
        [-n                     | --noecho]
        [(-l logfile            | --log logfile)]
        (-q "querystring"       | --query "querystring")
        (((-a "timewindow"      | --active "timewindow") |
          (-u period "timewindow" [updates] |
            --update period "timewindow" [updates]) |
          [-e | --extra])
          (-h                   | --historical))
        ((-x schemafile         | --xml schemafile) |
          (-d ["delimiter"]     | --delimited "delimiter"))
        (-p port                | --port port)
        (-o resultfile          | --output resultfile)
        server
Data API Utility completed.

C:\>
```

Saving the userid and password

This must be done before any of the following scenarios. Without this, the Utility can not make contact with the web service.

```
C:\>DataExportUtility -s username
Starting Data API Utility...
password:
Data API Utility completed.
C:\>
```

Getting historical data

This is the simplest query you can make. This example returns a delimited text result. The **`localhost`** parameter means that the Utility is running from the host where the report subsystem is installed. If the Utility is running from a different host than the Report Framework, use the hostname or IP address of the Report Framework instead of **`localhost`**.

```
C:\>DataExportUtility -q "select * from dcProperty" -h -d ":" localhost
Starting Data API Utility...
dcservice_id:service_name:name:value
10:fwddisplaynames                         :direct:true
10:fwddisplaynames                         :fwdbuff:0
10:fwddisplaynames                         :ksize:2000000
10:fwddisplaynames                         :maxsize:2000000
10:fwddisplaynames                         :minsize:500
10:fwddisplaynames                         :recorderhosts:montreal
10:fwddisplaynames                         :sources:1
10:fwddisplaynames                         :storage:data/forwarder
11:dm                                      :CONTEXT:0
11:dm                                      :SOURCE_ID:1
Data API Utility completed.

C:\>
```

Getting real-time updates

This real-time example shows how to get 3 updates of the **agentstate** table, 5 seconds of pause between each update, for the **D1** time window.

```
C:\>DataExportUtility -q "select agentname, agentstatename from agentstate" -u 5
 "D1" 3 -d "," localhost
Starting Data API Utility...
AGENTNAME,AGENTSTATENAME,B,D1
AGENTNAME,AGENTSTATENAME,B,D1
AGENTNAME,AGENTSTATENAME,B,D1
Data API Utility completed.
C:\>
```

Specifying data output and logs location

In this example, the log file is **`c:\log.txt`** instead of the default location of **`%pabase%/data/log/DataExportUtility/output.log`**. The **`-o`** option specifies that the query results are written to **`out.txt`** in the current directory.

```
C:\>DataExportUtility -l c:\log.txt -o out.txt -q "select * from dcProperty" -h
-d ":" localhost
Starting Data API Utility...
Data API Utility completed.

C:\>
```

Data API Utility query limits

Access to data in OA is provided using the SQL query language. The limits of the SQL support are discussed here as they relate to both the real-time and the historical data store.

This section includes the following topics:

- Consolidated data on page 24
- Real-time SQL limitations on page 25
- Data Set Characteristics on page 26
- OA historical data on page 27

Consolidated data

OA R6.1 includes a feature to provide a consolidated view of data in the event that a system is configured with multiple real-time subsystems. This means that if a single agent or service class exists in more than one real-time data store, the result set returned from a query contains a consolidated the view of the data from the separate data stores.

For example, imagine that there is an installation with two real-time subsystems, subsystem "A" and subsystem "B". There is work going on in both A and B for service class "Gold". In real-time subsystem "A" there were 40 workitems forwarded and in subsystem "B" there were 60 workitems forwarded. A query to get the wkforwardedCnt for the "Gold" service class from the svcClassSummary table is:

```
SELECT wkForwardedcnt from svcClassSummary where svcclassid =
'Gold';
```

The result of this query returns a wkForwardedCnt of 100.

Real-time SQL limitations

The spirit behind the SQL support for the real-time subsystem is to do the following:

- Prevent destructive types of queries. For example, DELETE, UPDATE, INSERT, DROP TABLE, CREATE TABLE on any rows, columns or tables. The SQL support that is provided allows for READ-ONLY access to data.
- Prevent queries that are computationally intensive and performance hindering. For example: ORDER BY, GROUP BY, aggregates (SUM, AVG, COUNT, MAX, MIN), sorting: UPPER,LOWER and nested queries.

Real time data can be accessed using a basic **SELECT** statement with a **WHERE** clause.

The following are not supported:

- The "**AS**" keyword
- "**SELECT ***"

There is support for table joins, as well as support for using table aliases. Whenever there are multiple tables in the **FROM** clause, all columns in the **SELECT** statement must be qualified with a table alias.

Example

Note:

The table schema below is only documented here as an example, it is not intended to represent the table schema of an actual table in OA.

Student table schema

```
StudentId (PK)
StudentName
```

Enrollment table schema

```
CourseId (PK)
StudentId (PK)
```

Correct query statement:

```
Select a.StudentId, a.StudentName, b.CourseId from Student a,
  Enrollment b where a.StudentId = b.StudentId;
```

Incorrect query statement:

```
Select a.StudentId, StudentName, CourseId from Student a,
  Enrollment b where a.StudentId = b.StudentId;
```

Even though the **CourseId** and **StudentName** columns are not present in both tables, the table alias still needs to be used in the **Select** statement.

Data Set Characteristics

An SQL query written against the real-time data store is represented as a **DataSet**. A unique DataSet is defined by three different characteristics.

1. The SQL query statement - this can be any SQL SELECT statement that adheres to the rules mentioned above.
2. The Time Window - this refers to one of the five different time windows administered for the real-time subsystem. There are five different time windows, which are **B**, **D1**, **D2**, **D3**, and **D4**. **B** refers to the base interval. The base interval represents data that is accumulated for the current half hour. At the end of each half hour interval, the data is sent to the Historical subsystem and reset to zero in the real-time subsystem. **D1** - **D4** represent four different daily intervals that can be administered to represent any twenty-four hour period of time. Just as the base interval data is cleared every half hour, the daily intervals are cleared every twenty-four hours. The daily interval data is not sent to historical at the end of each interval.
3. The Update Period - this represents the frequency with which the query should be refreshed against the real-time data store.

Note:

When a Time Window is selected, the real-time subsystem automatically appends the appropriate prefix to each of the interval columns specified in the **SELECT** statement. You should not append the Time Window prefix on any column in the **Select** query statement since this is done automatically.

Here is an example of how the **SQL SELECT** statement is internally modified by the real-time subsystem to return data for the correct Time Window.

SQL SELECT statement:

```
Select agentlogin, onbreakdur from agentstate
```

TimeWindow:

```
D4
```

Modified Statement:

```
Select agentlogin, D4_onbreakdur from agentstate
```

The **SELECT** statement above only has one interval column, **onbreakdur**. The statement is modified to get the data for the correct Time Window, the **D4** daily interval column. Since **agentlogin** is not an interval column the prefix is not appended to the column name. Notice that the initial SQL statement did not have the Time Window prefix on the **onbreakdur** column. The Time Window prefix should never be put on any column that is submitted to real-time in a SQL query.

OA historical data

The OA historical data store can contain data from multiple real-time sources. The data within the store is partitioned by **sourceID**. Data within the store can be viewed in two different manners:

1. SQL queries can be written to provide a historical view of data in the historical "h" tables from a single source. The "h" tables are tables that contain interval data. An example of an "h" table is the hSvcClassSummary table. Getting data from a single sourceID can be accomplished by including a valid **sourceID** in the **where** clause of the SQL statement.
2. SQL queries can be written to provide a consolidated view of data in the historical "h" tables across all real-time sources. Getting a consolidated view of the data can be accomplished by including a **sourceID** of `0` (zero) in the **where** clause of the SQL statement. The **sourceID** of `0` is a reserved **sourceID** that is never to represent an actual real-time source; it always represents the consolidated view as it relates to historical data in the "h" tables.

Examples:

To get data from the hSvcClassSummary table that represents data for a single source with a **SourceID** of **1**:

```
SELECT SvcClassID, AbandAheadCnt FROM hSvcClassSummary WHERE
  ContainerID = 1 AND RowDate = 2003051412

AND StartTime = 0

AND SvcClassID = '12'

AND ResourceMgrID = N'2'

AND SourceID = 1;
```

To get data from the hSvcClassSummary table that represents a consolidated view of the data across all sources:

```
SELECT SvcClassID, AbandAheadCnt FROM hSvcClassSummary WHERE
  ContainerID = 1 AND RowDate = 2003051412

AND StartTime = 0

AND SvcClassID = '12'

AND ResourceMgrID = N'2'

AND SourceID = 0;
```

Notice that the only difference between the two SQL statements shown above is the value used to represent the **SourceID**. The fist example has a **SourceID** of **1**; therefore the data is only for the real-time source that has a **SourceID** equal to **1**. The second statement has a **SourceID** of **0**; therefore the data is consolidated across all of the real-time sources.

The historical store does not have any SQL limitations outside those imposed by the chosen database vendor (e.g. Oracle, DB2, SQL Server).

Data Export API performance information

The Data Export API (API) used by the Utility is intended to support only a few simultaneous data-extracting applications due to OA system performance degradation caused by many simultaneous extractions. This is particularly true when accessing the real-time database.

For these reasons, this API is not suitable for supporting an agent desktop display since it would subject the system to a high volume of extractions. Displaying dashboards/statistics on agent desktops could use a similar, but differently optimized, API. The API could be modified/extended to pool the requests from multiple agents together before sending them to the database. This modified API would also be limited to extracting data from a small subset of the OA real-time database. These changes would significantly reduce the load on the OA real-time databases and allow OA to scale to the large number of agent desktops that are required.

The software doesn't limit or control the number of accesses through the API. You must apply judgement and you may need to adjust or limit accesses via the API to get acceptable performance for API extractions and reports.

Performance testing verifies that:

- Historical data: the API can export 8 database items for each of 5000 service classes every 30 minutes with a response time no greater than 5 minutes.
- Real-time data: the API can export 9 database items for each of 5000 agents every 30 seconds with a response time no greater than 10 seconds.

You can configure the maximum number of rows that the API returns. This prevents very large queries from affecting performance of a live OA system.

The default maximum number of rows for both the real-time and historical databases is set to 5000 rows. The defaults are both configurable by editing properties files as described below.

For real-time queries:

File: **`PABASE/data/admin/realtime/rt.properties`**

Entry: **DATASET_ROW_LIMIT**

Component that needs to be restarted: **DataServer**

How to restart: **`pa off dsvr, pa on dsvr`**

⚠ **CAUTION:**

Real-time reports are interrupted when the DataServer is turned off

For historical queries:

File on Windows platforms:
`PABASE\stumbras\webapp\WEB-INF\config\DataExport\DataExportService_Config.properties`

File on Solaris:
`/usr/iplanet/servers/https-stumbras/webapp/WEB-INF/config/DataExport/DataExportService_Config.properties`

File on AIX:
`/usr/WebSphere/AppServer/installedApps/BISCOTTI/OAReports.ear/stumbras.war/WEB-INF/config/DataExportDataExportService_Config.properties`

Entry: **MaxNumberOfRows**

Component that needs to be restarted: **Report Framework (Stumbras)**

How to restart: Follow instructions to start or stop the Report Framework in *Avaya Operational Analyst Release 6.1 Maintenance and troubleshooting*.

⚠ **CAUTION:**

All reports are interrupted when restarting the Report Framework

Index

A

Availability of this document 7

C

Consolidated data 24

D

Data API performance information 29
Data API Utility 9
Data API Utility query limits. 23
Data model 11
Data Set Characteristics 26

G

Getting historical data 20
Getting real-time updates 21
Getting the usage instructions 19

I

Installation 9
Intended users of this document 6

O

OA historical data 27
Overview of this document 6

P

Parameter substitution 18
Parameters 14
Parameters explained 15
Purpose of this document 6

R

Real-time SQL limitations 25
Reasons for reissuing this document 7
Related documentation 8

S

Safety labels 7
Saving the userid and password. 20
Security 10
Specifying data output and logs location 21

U

Usage examples 19
User Authentication 10
User authentication troubleshooting 10
Using the Data API Utility 13