New Features in Sybase® Avaki® EII 7.1 (Data Federation)

Document ID: DC00612-01-0710-01
Last revised: February 8, 2007

This document describes enhancements and new features included in Sybase Avaki EII 7.1, Data Integration (DI) Suite 1.1 Data Federation, and Sybase WorkSpace 1.7 Data Federation. For details about supported platforms, defect fixes, upgrading to the new release, and other product information, refer to the *Release Bulletin Sybase Avaki EII 7.1 (Data Federation)*. The release bulletin is available at http://infocenter.sybase.com—click the Avaki EII 7.1 link in the left pane, then select the Avaki 7.1 release bulletin.

Note Sybase® is changing the name of its EII offering from *Avaki* to *Data Federation*. Both names appear in the user interfaces and in the documentation, but they refer to the same product.

Avaki SQL engine enhancements

In Sybase Avaki® EII you can use SQL in two ways, and the two methods have different syntax requirements. You can execute a SQL query either against the Avaki virtual database or against a back-end (external) database.

Executing SQL queries against the Avaki virtual database

In this case, your SQL statement must refer to tables that have been explicitly exposed as Avaki SQL views. Your SQL statement is parsed and processed by Avaki's internal SQL engine (although in some cases, fragments of the query may be pushed down to back-end databases). Because your SQL statement is parsed by Avaki, it must conform to Avaki's SQL syntax, which is a subset of SQL/92. The SQL enhancements in release 7.1 apply to this Avaki SQL dialect.

There are two ways to execute a query against the Avaki virtual database:

- Via a JDBC client connected to Avaki using the Avaki native JDBC driver, the Sybase Jconnect JDBC driver, or the Sybase ODBC driver, or
- Via a virtual database operation that contains the query to be executed.

For details on connecting to Avaki via JDBC or ODBC, see the see the *Sybase Avaki EII API Guide*. For details on using virtual database operations, see the *Sybase Avaki EII Provisioning and Advanced Data Integration Guide*.

Executing SQL queries against a back-end database

There are two ways to execute a SQL query against a back-end database:

- Use a JDBC ad-hoc query
  In this case, you establish a connection from your JDBC client to Avaki in ad-hoc mode, specifying an Avaki database connector as a connection property. All SQL executed through this connection is passed directly to the specified database without being processed by Avaki. For details on using JDBC in ad-hoc mode, see the *Sybase Avaki EII API Guide*.
- Use a database operation (non-virtual) to execute a query against an Avaki database connector
  In this case, Avaki passes the entire query to the back-end database specified by the database connector; Avaki does not attempt to parse the query. Your query must

conform to the SQL syntax supported by the back-end database; it does not need to conform to the SQL syntax supported by Avaki's virtual database.

Note that you can expose a database operation as a table in Avaki's virtual database. When you query the table, two levels of SQL processing are performed. Suppose your database operation executes query Q1 against a given database connector, and you've exposed the database operation as a SQL view called T. Now you connect a JDBC client to Avaki and execute the query `select * from T`. This is what happens:

1. The Avaki SQL engine parses the statement `select * from T`—it is important that this statement conform to Avaki's SQL dialect.
2. The Avaki SQL engine obtains the contents for table T by executing the underlying database operation. When the database operation runs, Avaki sends query Q1 to the back-end database. Q1 must conform to the SQL syntax of the back-end database, but need not conform to Avaki's own SQL dialect.

SQL enhancements in this release

The SQL enhancements described in this section include support for a selection of aggregate, date, and numeric functions, as well as NOT IN predicates, HAVING clauses, and FULL OUTER JOIN clauses.

In Data Federation Studio/Avaki Studio, these SQL enhancements affect virtual database operations, which can serve as data sources for view models.

Note This section is not intended to be a full description of all the SQL statements that Avaki supports.

Aggregate functions

Aggregate functions summarize data over a group of rows from the database. The groups are formed using the GROUP BY clause or the WHERE clause of the SELECT statement. Aggregate functions are allowed only in the select list and in the HAVING and ORDER BY clauses of a SELECT statement.

COUNT(* | [ALL | DISTINCT] scalar-expression) Function

This function counts the number of rows in a group depending on the specified parameters. Sybase Avaki EII 7.1 adds support for COUNT (DISTINCT { expression | column-name }); previously, COUNT(*) and COUNT(expression) were supported.

Syntax

```
COUNT ( * | expression | DISTINCT { expression | column-name } )
```

`*`	Returns the number of rows in each group.
`expression`	Returns the number of rows in each group where the expression is not the null value.
`DISTINCT expression or column-name`	Returns the unique values in the expression, or the column with name column-name.

Usage notes

Rows where the value is the NULL value are not included in the count.

Use the DISTINCT function only when you have unique values in your columns. If the column specified in the DISTINCT function has a high number of non-unique values, you might receive a HashSet error, which indicates that not enough memory is available in the Java virtual machine (JVM) to complete processing. If you do need to increase the JVM heap limit, edit the Xmx value in this line of the <install-dir>\jboss\bin\run.bat file as needed:

```
set VM_ARGS=-Xms64m -Xmx256m
-Dsun.rmi.transport.connectionTimeout=120000
```

Standards and compatibility

SQL/92	Compatible
SQL/99	Vendor extension
Sybase	Compatible with Adaptive Server Enterprise and SQL Anywhere

Example

The following statement returns number of unique states in the employee table:

```
SELECT COUNT(DISTINCT state) FROM employee GROUP BY city
```

MAX() for non-numeric types

This function returns the maximum expression value found in each group of rows. Avaki EII 7.1 adds support for non-numeric types; previously, MAX() was supported only for numeric types.

Syntax

```
MAX ( expression | DISTINCT column name )
```

`expression`	The expression for which the maximum value is to be calculated. This is commonly a column name.
`DISTINCT column-name`	Returns the same as MAX(expression), and is included for completeness.

Usage notes

Rows where expression is NULL are ignored. Returns NULL for a group containing no rows.

Typically, the results obtained from the MAX function conform to the descriptions in “Executing SQL queries against the Avaki virtual database” on page 2 and “Executing SQL queries against a back-end database” on page 2. However, if a SELECT statement using the MAX function is used against a provisioned Avaki database and it does not contain a GROUP BY clause, then the query is processed as if it was submitted against a back-end database. If a SELECT statement using the MAX function is used

against a provisioned Avaki database and it **does** contain a GROUP BY clause, then the query is processed by Avaki's internal SQL engine; Avaki obtains the results by comparing the Unicode values of each character, one by one. (Note that a character's Unicode value varies depending on whether it is upper-case or lower-case.)

Standards and compatibility

SQL/92	Compatible
SQL/99	Core feature
Sybase	Compatible with Adaptive Server Enterprise and SQL Anywhere

Example

The following statement returns the "maximum" (or highest) last name value in the employee table.

```
SELECT MAX( emp_lname ) FROM employee
```

The following statement returns the "maximum" (or highest) birth date value in the employee table.

```
SELECT MAX( birth_date ) FROM employee
```

MIN() for non-numeric types

This function returns the minimum expression value found in each group of rows. Avaki EII 7.1 adds support for non-numeric types; previously, MIN() was supported only for numeric types.

Syntax

```
MIN ( expression | DISTINCT column name )
```

`expression`	The expression for which the minimum value is to be calculated. This is commonly a column name.
`DISTINCT column-umn-name`	Returns the same as MIN(expression), and is included for completeness.

Usage notes

Rows where expression is NULL are ignored. Returns NULL for a group containing no rows.

Typically, the results obtained from the MIN function conform to the descriptions in "Executing SQL queries against the Avaki virtual database" on page 2 and "Executing SQL queries against a back-end database" on page 2. However, if a SELECT statement using the MIN function is used against a provisioned Avaki database and it does not contain a GROUP BY clause, then the query is processed as if it was submitted against a back-end database. If a SELECT statement using the MIN function is used against a provisioned Avaki database and it **does** contain a GROUP BY clause, then the query is processed by Avaki's internal SQL engine; Avaki obtains the results by comparing the Unicode values of each character, one by one. (Note that a character's Unicode value will vary depending on whether it is upper-case or lower-case.)

Standards and compatibility

SQL/92	Compatible
SQL/99	Core feature
Sybase	Compatible with Adaptive Server Enterprise and SQL Anywhere

Example

The following statement returns the minimum (or lowest) first name value in the employee table.

```
SELECT MIN( emp_fname ) FROM employee
```

The following statement returns the minimum (or lowest) start date value in the employee table.

```
SELECT MIN( start_date ) FROM employee
```

Date functions

Date functions perform operations on date data types or return date information.

Date parts reference table

Many of the date functions use dates built from date parts. The following table shows the date parts and their possible values.

Date Part	Abbreviation	Values/Notes
Year	yy	1–9999
Quarter	qq	1–4
Month	mm	1–12
Week	wk	1–54; weeks begin on Saturday
Day	dd	1–31
Dayofyear	dy	1–366
Weekday	dw	1–7 (Sunday=1, ... , Saturday=7)
Hour	hh	0–23
Minute	mi	0-59
Second	ss	0–59
Millisecond	ms	0–999

GETDATE() function

This function returns the current year, month, day, hour, minute, second, and fraction of a second. The accuracy is limited by the accuracy of the system clock.

Syntax

```
GETDATE ()
```

Usage notes

The data type returned by this function is a TIMESTAMP.

The returned value will be based on the system clock of the connected grid server.

For a given SQL statement execution, all references to the GETDATE() function will be based on a single reading of the clock. For example, given the SQL statement:

```
SELECT GETDATE(), ... FROM ... WHERE colx > GETDATE()
```

The clock will be read a single time such that both references to the GETDATE() function result in exactly the same value.

Standards and compatibility

SQL/92	Vendor extension
SQL/99	Vendor extension
Sybase	Compatible with Adaptive Server Enterprise and SQL Anywhere. Both ASE and SQL Anywhere return the value as a DATETIME data type, and a JDBC type of TIMESTAMP.

Example

The following statement returns the system date and time:

```
SELECT GETDATE( )
```

The following examples show other possible uses of the GETDATE function:

```
SELECT GETDATE( ) WHERE 1 = 1
SELECT GETDATE( ), birth_date FROM employee
```

DATEPART() function

This function returns the value of part of a datetime value.

Syntax

```
DATEPART ( date-part, date-expression )
```

`date-part`	The date-part to be returned. For a complete listing of allowed date-parts, see the "Date parts reference table" on page 8.
`date-expression`	The date for which the part is to be returned. The date must contain the date-part field.

Standards and compatibility

SQL/92	Transact-SQL extension
SQL/99	Vendor extension
Sybase	Compatible with Adaptive Server Enterprise and SQL Anywhere

Example

The following statement returns the value 5.

```
SELECT datepart( month , '1987/05/02' )
```

DATEDIFF() function

This function returns the interval between two dates.

Syntax

```
DATEDIFF ( date-part, date-expression1, date-expression2 )
```

`date-part`	Allowable values: Year | quarter | month | week | day | hour | minute | second | millisecond Specifies the date-part in which the interval is to be measured. For more information about date-parts, see the "Date parts reference table" on page 8.
`date-expression1`	The starting date for the interval. This value is subtracted from date-expression2 to return the number of date-parts between the two arguments.
`date-expression2`	The ending date for the interval. Date-expression1 is subtracted from this value to return the number of date-parts between the two arguments.

Usage notes

This function calculates the number of date parts between two specified dates. The result is a signed integer value equal to (date2 - date1), in date parts.

DATEDIFF results are truncated, not rounded, when the result is not an even multiple of the date part.

When you use *day* as the date part, DATEDIFF returns the number of midnights between the two times specified, including the second date but not the first.

When you use *month* as the date part, DATEDIFF returns the number of first-of-the-months between two dates, including the second date but not the first.

When you use *week* as the date part, DATEDIFF returns the number of Sundays between the two dates, including the second date but not the first.

For the smaller time units, the following overflow values apply:

- milliseconds – 24 days
- seconds – 68 years

- minutes – 4083 years
- others – No overflow limit

The function returns an overflow error if you exceed these limits.

Standards and compatibility

SQL/92	Transact-SQL extension
SQL/99	Transact-SQL extension
Sybase	Compatible with Adaptive Server Enterprise and SQL Anywhere

Example

The following statement returns 1:

```
SELECT datediff( hour, '4:00AM', '5:50AM' )
```

The following statement returns 102:

```
SELECT datediff( month, '1987/05/02', '1995/11/15' )
```

The following statement returns 0:

```
SELECT datediff( day, '00:00', '23:59' )
```

The following statement returns 4:

```
SELECT datediff( day, '1999/07/19 00:00', '1999/07/23 23:59' )
```

The following statement returns 0:

```
SELECT datediff( month, '1999/07/19', '1999/07/23' )
```

The following statement returns 1:

```
SELECT datediff( month, '1999/07/19', '1999/08/23' )
```

DATEADD() function

This function returns the date produced by adding a number of the date parts to a date.

Syntax

```
DATEADD ( date-part, numeric-expression, date-expression )
```

`date-part`	Allowable values: year | quarter | month | week | day | hour | minute | second | millisecond The date-part to be added to the date. For more information about date-parts, see the "Date parts reference table" on page 8.
`numeric-expression`	The number of date-parts to be added to the date. The numeric-expression can be any numeric type, but the value is truncated to an integer.
`date-expression`	The date to be modified.

Standards and compatibility

SQL/92	Vendor extension
SQL/99	Vendor extension
Sybase	Compatible with Adaptive Server Enterprise and SQL Anywhere

Example

The following statement returns the value: 1995-11-02 00:00:00.0.

```
SELECT dateadd( month, 102, '1987/05/02' )
```

DATENAME() function

Returns the name of the specified part (such as the month “June”) of a datetime value, as a character string. Support for DATENAME() provides compatibility with ASE and SQL Anywhere.

Syntax

```
DATENAME ( date-part, date-expression )
```

`date-part`	The date-part to be named. For a complete listing of allowed date-parts, see the “Date parts reference table” on page 8.
`date-expression`	The date for which the date-part name is to be returned. The date must contain the requested date-part.

Usage notes

DATENAME returns a string, even if the result is numeric, such as 23 for the day.

Standards and compatibility

SQL/92	Transact-SQL extension
SQL/99	Vendor extension
Sybase	Compatible with Adaptive Server Enterprise and SQL Anywhere

Example

The following statement returns the value May.

```
SELECT datename( month , '1987/05/02' )
```

Numeric functions

Numeric functions perform mathematical operations on numerical data types or return numeric information.

TRUNCNUM() function

This function truncates a number at a specified number of places after the decimal point. Support for TRUNCNUM() provides compatibility with SQL Anywhere.

Note This function is the same as TRUNCATE, but does not cause keyword conflicts.

Syntax

```
TRUNCNUM ( numeric-expression, integer-expression )
```

`numeric-expression`	The number to be truncated.
`integer-expression`	A positive integer specifies the number of significant digits to the right of the decimal point at which to round. A negative expression specifies the number of significant digits to the left of the decimal point at which to round.

Standards and compatibility

SQL/92	Vendor extension
SQL/99	Vendor extension
Sybase	Compatible with SQL Anywhere; not supported in Adaptive Server Enterprise.

Example

The following statement returns the value 600.

```
SELECT TRUNCNUM( 655, -2 )
```

The following statement returns the value 655.34.

```
SELECT TRUNCNUM( 655.348, 2 )Example
```

Other SQL language clauses and expressions

This section describes other enhancements to SQL language clauses and expressions.

NOT IN predicate

Previously, Avaki supported the “IN” predicate. Adding support for “NOT IN” improves compatibility with ASE and SQL Anywhere.

Syntax

The syntax for IN conditions is as follows:

```
expression [ NOT ] IN { ( subquery ) | ( expression2 ) | (
  value-expr, . . . ) }
```

An IN condition, without the NOT keyword, evaluates according to the following rules:

- TRUE if expression is not NULL and equals at least one of the values.
- UNKNOWN if expression is NULL and the values list is not empty, or if at least one of the values is NULL and expression does not equal any of the other values.
- FALSE if expression is NULL and subquery returns no values; or if expression is not NULL, none of the values are NULL, and expression does not equal any of the values.

The NOT keyword interchanges TRUE and FALSE.

The search condition expression IN (values) is identical to the search condition expression = ANY (values). The search condition expression NOT IN (values) is identical to the search condition expression <> ALL (values).

The value-expr arguments are expressions that take on a single value, which may be a string, a number, a date, or any other SQL data type.

Standards and compatibility

IN conditions are compatible with Adaptive Server Enterprise and SQL Anywhere.

HAVING clause

Avaki EII 7.1 adds support for the HAVING clause for compatibility with ASE and SQL Anywhere.

Syntax

```
SELECT [ ALL | DISTINCT ] [ row-limitation ] select-list

[ INTO { hostvar-list | variable-list | table-name } ]

[ FROM table-expression ]

[ WHERE search-condition ]

[ GROUP BY [ group-by-expression ]

[ HAVING search-condition ]
```

The HAVING clause selects rows based on the group values and not on the individual row values. The HAVING clause can only be used if either the statement has a GROUP BY clause or the select list consists solely of aggregate functions. Any column names referenced in the HAVING clause must either be in the GROUP BY clause or be used as a parameter to an aggregate function in the HAVING clause.

When GROUP BY is used, the HAVING clause can reference only identifiers named in the GROUP BY clause. The exception is that the select-list and HAVING clause may contain aggregate functions.

The search-condition serves basically the same purpose and has the same syntax as the search-condition for the WHERE clause, with the implications described above.

Example

The following is an example of a query containing a HAVING clause:

```
SELECT stor_id, AVG(qty) FROM salesdetail GROUP BY stor_id
HAVING AVG(qty) > 200
```

FULL OUTER JOIN expression

Avaki EII 7.1 adds support for the FULL OUTER JOIN clause.

In a full outer join, all rows are preserved for both of the tables, and nulls are supplied for rows that do not satisfy the join condition.

```
<join-type> ::= <table reference> FULL OUTER JOIN <table
  reference>  ... ON <conditional expression>
```

Example

This section describes an example of a full outer join. Suppose the following tables exist:

Table A

W	X
A	11
B	12
C	13

Table B

Y	Z
A	21
C	22
D	23

The following select statement and result table show how the statement is processed.

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
```

Result table:

W	X	Y	Z
A	11	A	21
C	13	C	22
<null>	<null>	D	23
B	12	<null>	<null>

SySAM license management

You must use the Sybase Software Asset Management (SySAM) licensing system to administer Sybase Avaki and Data Federation licenses. The method you use to set up SySAM varies slightly, depending on the operating system on which you are installing Avaki/Data Federation.

- For HP-UX environments, you must install SySAM into the same directory as Avaki and perform some manual setup tasks. For instructions, see "Setting up license management for Avaki on HP-UX" on page 19.
- For all other operating systems, when you install Data Integration (DI) Suite, SySAM is installed automatically; Avaki is installed if you select the Data Federation component. For more information, see "Setting up license management in Data Integration Suite" on page 22.

Refer to the SySAM documentation for more information. The following SySAM documentation is included with Sybase Avaki in the <Avaki-install-dir>\docs directory and on the Sybase Technical Library Product Manuals Web site.

- *Sybase Software Asset Management User's Guide*
- *FLEXnet Licensing End User Guide*
- *SAMreport Users Guide*
- *SySAM Quick Card* (available as a PDF file in the <Avaki-install-dir>\docs directory only)

Setting up license management for Avaki on HP-UX

For HP-UX environments, you must install SySAM to the same root directory where you installed Avaki and you must manually place Avaki license files in the correct directory. If you want to use the SySAM served license model, you must also install SySAM on the machine where you want the SySAM server to run.

Follow these steps to prepare for license management.

Step 1 Become familiar with the setup requirements of SySAM. For more detailed information about SySAM, see the *Sybase Software Asset Management User's Guide*.

Step 2 Make sure you have all necessary installation media.

If you are installing from CDs, you should have two CDs:

- A Sybase Avaki installation CD for HP-UX PA-RISC or HP-UX Itanium. This CD contains all components except Avaki Studio, and includes SySAM 2.0.
- A Sybase Avaki installation CD for MS Windows. You will use this CD to install only the Avaki Studio component.

If you are downloading the product from the Sybase Product Download Center (SPDC) at *https://sybase.subscribenet.com*, download all Sybase Avaki components available for your operating system, plus the MS Windows Avaki Studio component, and SySAM 2.0.

Step 3 Install Sybase Avaki.

From the HP Sybase Avaki CD, install all necessary components for the machine on which you are installing Avaki.

From the Windows CD, install only Avaki Studio.

For more installation and configuration information, see the *Sybase Avaki EII Administration Guide*, Release 7.0.

Note Do not install Avaki to the root ("/") directory.

Step 4 Install SySAM.

The installer files are located here:

- Extras/SYSAM_HP_INSTALL/hpia64 for HP-UX Itanium
- Extras/SYSAM_HP_INSTALL/hppa64 for HP-UX PA-RISC

Install SySAM to the same level directory into which you installed Avaki. For example, if you install Sybase Avaki into /Sybase/DF-7_1, then install SySAM 2.0 into /Sybase. (SySAM silently adds the SYSAM-2_0 directory.)

Your resulting directory structure should look similar to the following:

```
/Sybase/DF-7_1
/Sybase/SYSAM-2_0
```

Step 5 Verify that the directory <install_dir>/SYSAM-2_0/licenses exists.

The SySAM installer creates this directory automatically.

Step 6 If you are using the SySAM served license model and you do not want to run the license server on the same machine as the Avaki server machine, install SySAM onto your license server machine.

Step 7 Use the appropriate command to obtain the host ID for your system.

The format of the command varies according to your operating system. For more information, see the *FLEXnet Licensing End User Guide*.

- For the unserved license model - run the command from the <install_dir>/SYSAM-2_0/bin directory.
- For the served license model - run the command from the license server's SYSAM-2_0/bin directory.

You will use the host ID when generating licenses at the Sybase Product Download Center (SPDC).

The value for your host ID will look similar to one of these examples:

- 778DA450 or #2005771344 on HP-UX PA-RISC platforms
- ID_STRING=9c766319-db72-d411-af62-0060b05e4c05 on HP-UX Itanium 64-bit.

Step 8 Go to the Sybase Product Download Center (SPDC) at *https://sybase.subscribenet.com*.

Step 9 Generate the Avaki licenses that you need.

For instructions, see the *Sybase Software Asset Management User's Guide*.

Step 10 For the unserved license model, place the generated Avaki license files into the directory <install_dir>/SYSAM-2_0/licenses.

The files can have any name, but must have a file type of **.lic**.

Step 11 For the served license model, place a file in the directory <install_dir>/SYSAM-2_0/licenses that identifies the license server.

This file can have any name but it must have a file type of **.lic** and contain two lines of content in the following format:

```
SERVER [ServerName] ANY
USE_SERVER
```

Where [ServerName] is the machine name of the license server.

Step 12 Complete the SySAM configuration process as described in the *Sybase Software Asset Management User's Guide*.

Note If you install Sybase Avaki without downloading license files, it will run successfully for a 30-day grace period; after this time, the Avaki server will shut down and will no longer start.

Setting up license management in Data Integration Suite

Sybase Avaki and SySAM are components of Data Integration (DI) Suite. When you install DI Suite, SySAM is installed automatically; Avaki is installed if you select the Data Federation component. You provide license information using the installation wizard.

You should become familiar with the setup requirements of SySAM before installing DI Suite. For more detailed information about SySAM, see the *Sybase Software Asset Management User's Guide*.

During installation, a SySAM License installer screen asks you to indicate where to find the Data Federation license. The information that you provide varies depending on whether you are using served or unserved licenses.

For specific instructions on installing DI Suite components, refer to the *Sybase Data Integration Suite Installation Guide*, Version 1.1.

Note You must have a license file in order to install Sybase Data Federation (and any other DI Suite component). DI Suite does not provide a 30-day grace period.

BLOB and CLOB support in SQL views

Sybase Avaki EII 7.1/Data Federation 1.1 provides preliminary support for BLOBs (binary large objects) and CLOBs (character large objects) in Avaki SQL views. (Avaki database operations and data services have always supported BLOBs and CLOBs.) Now you can:

- Create either provisioned or generated SQL views that contain BLOB and CLOB columns. (You can create and use SQL views in Avaki Studio/WorkSpace Data Federation Studio as well as in Avaki's web UI.)
- Access BLOB and CLOB data in SQL views via JDBC using the Avaki JDBC driver. (Neither the Jconnect driver nor ODBC will work for this purpose.)

This support is limited to *BLOB* and *CLOB* types only. That is, when Avaki communicates with a database via JDBC, Avaki must see a SQL *BLOB* or *CLOB* type, and not one of the other types sometimes used in place of *BLOB* or *CLOB*. For example,

- The following *BLOB* substitutes are not supported in Avaki:
  - the *image* type in Sybase ASE
  - the *bfile* type in Oracle
- The following *CLOB* substitutes are not supported in Avaki:
  - the *text* type in Sybase ASE
  - the *mediumtext* and *longtext* types in MySQL

You can access BLOBs and CLOBs via the Avaki JDBC driver using the same techniques you use to access data of other types. For details, see the *Sybase Avaki EII API Guide*.

The preliminary BLOB and CLOB support in this release does not include a streaming mechanism for BLOB or CLOB data. When a client (connecting to Avaki via the Avaki JDBC driver) requests a row containing a BLOB or CLOB, Avaki ships all of the data in the row, including the BLOB or CLOB, across the wire from the grid server to the client. It's not currently possible for the client to inspect part of the BLOB or CLOB or its containing row before requesting the rest. Consequently, performance will be poor when the client doesn't want to read all the data. However, in a case where the client always wants to read the entire BLOB or CLOB, we believe performance will be acceptable. For example, consider a scenario where the client wants to select BLOB or CLOB columns containing images to be displayed. In this case, the client needs to read all of the data to display the image properly, so the ability to request smaller chunks of the BLOB or CLOB is not crucial.

JRE upgrade

Avaki EII 7.1 includes an upgrade to Avaki's Java Runtime Environment (JRE). The JRE upgrade affects all Avaki servers and clients, including grid servers, grid domain controllers, share servers, proxy servers, data grid access servers, command line clients, and instances of Avaki Studio/WorkSpace Data Federation Studio.

The JRE upgrade addresses several issues, including the following:

- Change in Daylight Saving Time:
  In 2007, the algorithm governing the dates on which Daylight Saving Time begins and ends in the US and Canada is changing. Formerly DST began on the second Sunday in April and ended the last Sunday in October; beginning this year, DST will begin on the second Sunday in March and end the first Sunday in November. Applications running on older JREs might report incorrect time from March 11, 2007 through April 2, 2007 and from October 29, 2007 through November 4, 2007. For more on this issue, see

  http://java.sun.com/developer/technicalArticles/Intl/USDST/

- RSA/SSL security vulnerability:
  JRE and JSSE verify some incorrect RSA signatures. This allows some applets or applications that are signed by forged signing certificates and Web sites with forged Web server certificates to be verified as valid. For more on this issue, see

  http://sunsolve.sun.com/search/document.do?assetkey=1-26-102686-1

Note If you choose not to upgrade to Avaki 7.1, or if you don't upgrade right away, Sybase recommends that you upgrade the JREs of all the servers and clients in your Avaki 6.2, Avaki 7.0, or DI Suite 1.0 Data Federation domain.

The JRE upgrade procedure is available at http://infocenter.sybase.com—click the Avaki EII 7.0 link in the left pane, then select the Avaki 6.2/7.0 release bulletin.

Internal database upgrade

Avaki EII 7.1 includes an upgrade to Avaki's internal database that involves migration to a new file format. If you're upgrading to this release, you must perform a few preliminary manual steps on each grid server in your domain to trigger the automatic migration process. The steps are described in the release upgrade procedure in the *Release Bulletin Sybase Avaki EII 7.1 (Data Federation)*. (This document is available at http://infocenter.sybase.com—click the Avaki EII 7.1 link in the left pane, then select the Avaki 7.1 release bulletin.) Code built into this release completes the migration automatically when you restart your Avaki servers at the end of the upgrade procedure.

Internationalization improvements

Avaki EII 7.1 includes improvements to multibyte character support. For details, see the *Release Bulletin Sybase Avaki EII 7.1 (Data Federation)*. This document is available at http://infocenter.sybase.com—click the Avaki EII 7.1 link in the left pane, then select the Avaki 7.1 release bulletin.

Note The improvements in Sybase Avaki 7.1 do not constitute full internationalization of Avaki EII software.

Written by Emily Goodin and Beth Thoenen

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.