

---

# New and changed features for Unify ACCELL/SQL Release 9.1

This document lists and explains the features that have been added and behavior changes that have been introduced for Release 9.1 of Unify ACCELL/SQL. Some of these changes or enhancements may have been available in earlier releases or patches.

## Table of Contents

UPDATED ORACLE DATABASE ACCESS LAYER .....	4
<b>OSTCACHESZ</b> configuration variable no longer used.....	4
Support for Oracle <b>TIMESTAMP</b> columns .....	4
Support for <b>CLOB</b> and <b>BLOB</b> columns.....	4
ACCELL/SQL executables are linked against Oracle static libraries .....	4
NEW <b>DATETIME</b> DATA TYPE .....	5
<b>DATETIME</b> variables and database columns .....	5
ACCELL/SQL <b>DATETIME</b> variables and DataServer .....	5
ACCELL/SQL <b>DATETIME</b> variables and Oracle .....	5
Creating <b>DATETIME</b> variables.....	6
Input and output formatting of <b>DATETIME</b> values.....	6
Configuration variables for <b>DATETIME</b> support .....	8
<b>UDATETIMEFMT</b> .....	8
<b>MERIDIANFMT</b> .....	8
<b>DATETIMENULLCH</b> .....	8
New system functions for <b>DATETIME</b> support .....	9
to_datetime\$( ) .....	9
make_datetime\$( ).....	9
current_datetime\$( ) .....	10
datetime_to_mdym\$( ) .....	10
mdym_to_datetime \$( ) .....	11
Changed system functions for <b>DATETIME</b> support .....	12
to_num\$( ) .....	12
to_string\$( ) .....	12
to_string_using\$( ).....	12
Using <b>DATETIME</b> in <b>RPT</b> .....	12
Reading <b>DATETIME</b> values through the <b>INPUT</b> section.....	12
Printing <b>DATETIME</b> values.....	12
APACHE TOMCAT SERVLET/JSP ENGINE INSTALLED FOR USE WITH ACCELL/WEB AND RPT/WEB .....	13
Installation.....	13
Starting Tomcat .....	13
Stopping Tomcat .....	14
Deploying/Undeploying web applications.....	14
<b>RPTDIVIDEBYZERO</b> CONFIGURATION VARIABLE .....	15

NEW CONFIGURATION VARIABLE **ACL\_LIC\_WARN** ..... 16

---

## Updated Oracle database access layer

The ACCELL/SQL database access layer for Oracle had been updated to use the latest Oracle Call Interface API. This change provides a solid foundation for future enhancements to Oracle connectivity. Although the change should be transparent to existing applications, there are some differences which are described in the following sections.

---

## OSTCACHESZ configuration variable no longer used

The old Oracle database access layer maintained a cache of statements to avoid the performance impact of preparing statements during each execution. The size of this cache was calculated automatically but could be overridden by an undocumented configuration variable, **OSTCACHESZ**. As newer versions of Oracle include their own global statement cache, the ACCELL/SQL statement cache is redundant and has been removed. Therefore the **OSTCACHESZ** configuration variable no longer has any effect.

---

## Support for Oracle **TIMESTAMP** columns

ACCELL/SQL now supports database columns of type **TIMESTAMP**. These columns will map to variables of the new type **DATETIME**. For more details, see the section of this document that covers the new **DATETIME** ACCELL/SQL data type.

---

## Support for **CLOB** and **BLOB** columns

Oracle large object (LOB) column types **CLOB** and **BLOB** are now supported. The behavior of **CLOB** and **BLOB** columns is identical to the behavior of the already supported **LONG** and **LONG RAW** columns respectively.

---

## ACCELL/SQL executables are linked against Oracle static libraries

The ACCELL/SQL executables in the release (**CAMGR**, **AGEN**, etc.) are linked against Oracle static libraries. This makes it unnecessary to set the **LD\_LIBRARY\_PATH**, **SHLIB\_PATH** or **LIB\_PATH** (depending on the operating system) environment variables before starting these executables.

However, since Oracle does not install static libraries by default, custom ACCELL/SQL managers created with **amgr.ld** and custom **RPTs** created with **rpt.ld** will link against the Oracle shared libraries by default. If running such a custom manager, the **LD\_LIBRARY\_PATH**, **SHLIB\_PATH** or **LIB\_PATH** environment variables must include *\$ORACLE\_HOME/lib* for the executable to locate the Oracle shared libraries. If static linking of a custom manager or **RPT** is desired, first the static libraries need to be created. To do so, the Oracle system administrator must run the **genIntst** utility found in *\$ORACLE\_HOME/bin*. Once this is done, set the environment variable **ORACLE\_STATIC\_LINK** to **TRUE** in the environment before executing **amgr.ld** or **rpt.ld**.

---

## New DATETIME data type

ACCELL/SQL 9.1 introduces a new data type – **DATETIME**. The **DATETIME** data type can be used to store a moment in time with microsecond precision. **DATETIME** columns are stored in a time zone agnostic format and are converted to and from the current session’s time zone when displayed or input, and converted to and from the database storage format when stored or retrieved.

ACCELL/SQL **DATETIME** variables support dates after 1752 and through the year 9999. Attempts to use or create **DATETIME** values outside these limits will generate an overflow error. Note that the database used may not support all values in this range.

---

## DATETIME variables and database columns

**DATETIME** variables can be stored in and retrieved from database columns, either as target table variables or in SQL statements in the 4GL. DataServer **DATETIME** columns and Oracle **DATE** and **TIMESTAMP** columns are currently supported.

---

### ACCELL/SQL **DATETIME** variables and DataServer

DataServer **DATETIME** columns map directly to ACCELL/SQL **DATETIME** columns. A target variable tied to DataServer **DATETIME** columns will take the **DATETIME** ACCELL/SQL variable type. An untyped general variable used to select a **DATETIME** column in a 4GL **SELECT** statement will be set to the **DATETIME** data type. No other DataServer data type can be mapped to an ACCELL/SQL **DATETIME** variable.

Since DataServer **DATETIME** columns have only millisecond precision, ACCELL/SQL **DATETIME** columns being stored in a DataServer **DATETIME** column will be truncated to the nearest millisecond.

---

### ACCELL/SQL **DATETIME** variables and Oracle

ACCELL/SQL **DATETIME** variables can be mapped to Oracle **DATE** or **TIMESTAMP** columns, either as target variables or as return values from a 4GL **SELECT** statement.

If an Oracle **DATE** column is used the default ACCELL/SQL data type is **DATE**; this maintains compatibility with existing applications. This default may be overridden by selecting the **DATETIME** data type when creating a form field, by specifying the **DATETIME** data type in the **LOCAL** section of the form script for non-screen target fields, or by setting the data type of a general variable to **DATETIME** before executing an SQL statement in the 4GL.

Since Oracle **DATE** columns do not store fractions of seconds, any such portion of a **DATETIME** variable will be discarded when the value is stored. **DATETIME** values retrieved from Oracle **DATE** columns will never have a fractional second portion.

If an Oracle **TIMESTAMP** column is used, the data type of the mapped variable will be **DATETIME**. Unlike an Oracle **DATE** column, an Oracle **TIMESTAMP** column cannot be mapped to an ACCELL/SQL **DATE** or **TIME** variable. Oracle **TIMESTAMP** columns can have a fractional second portion and are declared as having fractional seconds precision from 0 (no fractional second portion) to 9 (nanosecond precision). The default precision for a **TIMESTAMP** column is 6 (microsecond precision). When a **TIMESTAMP** column with more than microsecond precision is fetched into an ACCELL/SQL **DATETIME** variable, any precision past microseconds is discarded. When an ACCELL/SQL **DATETIME**

variable is stored into an Oracle **TIMESTAMP** column with less than microsecond precision the excess precision will be discarded.

---

## Creating **DATETIME** variables

**DATETIME** variables can be created in the following ways:

- Screen fields can be specified as being of type **DATETIME** in the Add Field form. If the field is a target field, the data type of the underlying column must be compatible (DataServer **DATETIME** or Oracle **DATE** or **TIMESTAMP**).
- Non-screen target variables will be of type **DATETIME** if the underlying column is a DataServer **DATETIME** or Oracle **TIMESTAMP**.
- Non-screen target variables with an underlying Oracle **DATE** column will default to the ACCELL/SQL **DATE** data type. This default can be overridden by declaring the variable as being of type **DATETIME** in the **LOCAL** code section of the form script, for example:

```
FORM orders TARGET_TABLE orders
LOCAL
    $order_submitted DATETIME
...
```

- An untyped and undefined general variable can be set to type **DATETIME** by assigning a value of type **DATETIME**. Several functions (such as `to_datetime$( )` and `make_datetime$( )`) return **DATETIME** values and can be used for this purpose:

```
...
SET $var1 TO to_datetime$(NULL);
...
```

---

## Input and output formatting of **DATETIME** values

Developers can control the display and input format for **DATETIME** variables either through the use of the **UDATETIMEFMT** configuration variable, by specifying a value for the “Display Format” attribute when adding a **DATETIME** field to a form, or by setting the **DISPLAY\_FORMAT** field attribute in the form script. Values can also be formatted by using the `to_string$( )` or `to_string_using$( )` functions, or by specifying a format in the **USING** clause of the 4GL **DISPLAY** statement.

The format template for a **DATETIME** value may contain the following characters:

Character	Display result
<b>YY</b>	An Arabic numeral representing the year. The year can be expressed by two digits or four digits.
<b>YYYY</b>	For data entry, if a 2-digit year is specified, the century is determined by the <b>CENTURY_CUTOFF</b> external preference.
<b>M</b>	An Arabic numeral that represents the month. A single <b>M</b> displays numbers less than 10 without a leading zero; <b>MM</b> displays a leading zero when the month is less than 10, or example, 02.
<b>D</b>	An Arabic numeral representing the day. A single <b>D</b> displays a number less than 10 without a leading zero; <b>DD</b> displays a leading zero when the day is less than 10, for example, 09.

<b>H</b>	An Arabic numeral representing the hour. A single <b>H</b> displays numbers less than 10 without a leading zero; <b>HH</b> displays a leading zero when the hour is less than 10, for example, 02.
<b>HH</b>	The hour is based on a 12-hour clock if the meridian indicator ( <b>MI</b> , see below) is also specified; otherwise, the hour is based on a 24-hour clock.
<b>I</b>	An Arabic numeral representing the minute. A single <b>I</b> displays numbers less than 10 without a leading zero; <b>II</b> displays a leading zero when the minutes are less than 10, for example, 02.
<b>II</b>	
<b>SS</b>	An Arabic numeral representing the seconds.
<b>.S</b>	An Arabic numeral representing the fractional seconds (1 to 6 digits). If less than 6 digits of precision are specified, the value is truncated, not rounded.
<b>.SS</b>	
<b>.SSS</b>	During input, users may enter from 1 to 6 digits for the fractional seconds regardless of how many digits are specified for display.
<b>.SSSS</b>	
<b>.SSSSS</b>	
<b>.SSSSSS</b>	
<b>MI</b>	A character string representing the Meridian indicator. If this indicator is present, the H or HH specification is based on a 12-hour clock. The meridian indicator strings default to 'AM' and 'PM' and can be changed by the <b>MERIDIANFMT</b> configuration variable.
<i>trim</i>	The elements of a <b>DATETIME</b> can be separated by one or more <i>trim</i> characters, such as a slash (/), a dash (-), a dot (.), or a space ( ). These characters are input or output as is and can optionally be enclosed in single quotation marks in the format specification.

If the **SS.SSS** characters are at the end of the format string, both seconds and fractional seconds are optional during input.

The default value for **UDATETIMEFMT** is '**YYYY-MM-DD HH:II:SS.SSS**' which corresponds to the ISO 8601 standard.

The character specified by the **DATETIMENULLCH** configuration variable will be used to represent a null value. If the **DATETIMENULLCH** variable is not set, the default **NULLCH** variable will be used instead.

During input of a **DATETIME** form field, the ACCELL/Manager will first attempt to parse the value using the format specifically defined for the field ("Display Format" or **DISPLAY\_FORMAT**) if one is defined. If this fails, the ACCELL/Manager will attempt to parse the value using the format defined by the **UDATETIMEFMT** configuration variable.

The following table illustrates a variety of **DATETIME** format templates displaying the same **DATETIME** value. These examples assume that the **MERIDIANFMT** configuration variable does not override the default setting.

<b>Format</b>	<b>Display result</b>
YYYY-MM-DD HH:II:SS.SSS	2008-11-25 17:23:15.234
MM/DD/YY H:II:SSMI	11/25/08 5:23:15PM
DD/MM/YYYY HH:II:SS.SSSSSS	25/11/2008 17:23:15.234760
M/D/YY H:I:SS	11/25/08 17:23:15
MM'/'DD'/'YYYY HH:II:SS	11/25/2008 17:23:15

---

## Configuration variables for **DATETIME** support

---

### **UDATETIMEFMT**

Default format in which to accept and display **DATETIME** values.

Valid values: A combination of the **DATETIME** format template characters. The format template must be enclosed in quotation marks (*"format\_template"*).

Default value: **"YYYY-MM-DD HH:II:SS.SSS"**

Example: **"MM/DD/YY HH:II:SSMI"**

Additional help: *Input and output formatting of **DATETIME** values* on page 6.

---

### **MERIDIANFMT**

The **MERIDIANFMT** configuration variable specifies the string values to use for the meridian indicator. A meridian indicator is used for the display of **DATETIME** values only.

It can be a single string representing the afternoon value, or two strings separated by a slash (/) representing the morning value (prior to the slash) and the afternoon value (following the slash).

Valid values: A string enclosed in quotation marks.

Default value: **"AM/PM"**

Example: **"in the morning/in the afternoon"**

---

### **DATETIMENULLCH**

The **DATETIMENULLCH** configuration variable specifies the null display character for **DATETIME** values. If **DATETIMENULLCH** is set to **"\*"**, a default-formatted null **DATETIME** value is displayed as

\*\*\*\*\*.

If the **DATETIMENULLCH** configuration variable is not set, then the setting of **NULLCH** is used.

When a user executes the set null operation, the null display character for **DATETIME** data is determined by the value in this preference.

Valid values: Any printable character enclosed in quotation marks.

Default value: **"\*"**

Example: **"#"**

## New system functions for DATETIME support

The following system functions have been added to handle **DATETIME** variables:

---

### to\_datetime\$( )

	<i>System function</i>	<i>Datetime conversion</i>
<b>Syntax</b>	<b>to_datetime\$( value )</b>	
<b>Arguments</b>	<i>value</i>	( <b>NUMERIC/BINARY/DATE/DATETIME</b> ) The value to be converted.
<b>Return Values</b>	( <b>DATETIME</b> )	The <b>DATETIME</b> value of <i>value</i> .
<b>Description</b>	<p>The <b>to_datetime\$( )</b> system function converts a <i>value</i> to a <b>DATETIME</b> value. The <i>value</i> can be any <b>STRING</b>, <b>NUMERIC</b>, <b>BINARY</b>, <b>DATE</b> or <b>DATETIME</b> value in the range of <b>DATETIME</b> values valid in the database.</p> <p>If <i>value</i> is a <b>BINARY</b> value, its length is assumed to be compatible with the length of a <b>DATE</b> value.</p> <p>If <i>value</i> is a <b>NUMERIC</b>, it must be the number of microseconds since <b>January 1st, 1970 00:00:00.000000 GMT</b> (the epoch). Negative values represent dates and times prior to the epoch. Due to the large numeric values that may need to be passed in to this function to represent a <b>DATETIME</b> value, the <b>UNUMERIC64</b> configuration variable should be set to <b>TRUE</b> to allow ACCELL/SQL to support 64-bit integer values. Converting a <b>DATETIME</b> value to a numeric value using <b>to_num\$( )</b> and converting the numeric value back into a <b>DATETIME</b> value with <b>to_datetime\$( )</b> results in the same <b>DATETIME</b> value.</p> <p>If <i>value</i> is a <b>STRING</b>, it will be parsed according to format template specified by the <b>UDATETIMEFMT</b> configuration variable.</p> <p>The <b>to_datetime\$( )</b> function can accept a <b>DATETIME</b> argument; it returns the <b>DATETIME</b> value without converting it.</p> <p>If <i>value</i> is a null value, <b>to_datetime\$( )</b> returns a null <b>DATETIME</b> value.</p>	
<b>Example</b>	<p>The following <b>SET</b> statement uses <b>to_datetime\$( )</b> to set the <b>DATETIME</b> variable <b>dtm_val</b> to the <b>DATETIME</b> value of 1314148705000000. The <b>DATETIME</b> value is 2011-08-23 18:18:25.000GMT.</p> <pre>SET dtm_val TO to_datetime\$(1314148705000000)</pre> <p>The following code adds 1 day to a <b>DATETIME</b> variable:</p> <pre>SET dtm_val TO to_datetime\$(to_num\$(dtm_val) + 86400000000);</pre>	
<b>See also</b>	<b>date_to_mdy\$( )</b> , <b>mdy_to_date\$( )</b> , <b>make_datetime\$( )</b> , <b>null_convert\$( )</b> , <b>str_to_date\$( )</b> , <b>to_amount\$( )</b> , <b>to_binary\$( )</b> , <b>to_bool\$( )</b> , <b>to_date\$( )</b> , <b>to_float\$( )</b> , <b>to_num\$( )</b> , <b>to_string\$( )</b> , <b>to_text\$( )</b> , <b>to_time\$( )</b> , <b>val_to_str\$( )</b>	

---

### make\_datetime\$( )

	<i>System function</i>	<i>Date conversion</i>
<b>Syntax</b>	<b>make_datetime\$( date, time )</b>	
<b>Arguments</b>	<i>date</i>	( <b>NUMERIC/BINARY/DATE/DATETIME</b> ) The date value.
	<i>time</i>	( <b>NUMERIC/BINARY/TIME/DATETIME</b> ) The time value.
<b>Return Values</b>	( <b>DATETIME</b> )	The <b>DATETIME</b> combination of the <i>date</i> and <i>time</i> arguments.

**Description** The **make\_datetime\$( )** system function combines separate **DATE** and **TIME** values to a single **DATETIME** value. The *date* can be any **NUMERIC, BINARY, DATE** or **DATETIME** value in the range of **DATE** values valid in the database. The *time* can be any **NUMERIC, BINARY, TIME** or **DATETIME** value in the range of **TIME** values valid in the database.

If either *date* or *time* is a **BINARY** value, its length is assumed to be compatible with the length of a **DATE** or **TIME** value, respectively.

If the **DATE** value is a **DATETIME**, then only the **DATE** portion is used. The **TIME** portion is discarded. If the **TIME** value is a **DATETIME**, then only the **TIME** portion is used. The **DATE** portion is discarded.

If both *date* and *time* are NULL, **make\_datetime\$( )** will return a NULL **DATETIME** value. If only *date* or *time* is NULL, ACCELL/Manager will display an error and return **UNDEFINED**.

**Example** This **SET** statement uses **make\_datetime\$( )** to set the **DATETIME** variable **dtm\_val** to the combination of the **DATE** value of 200 and the time value of 572. The **DATETIME** value is 1990-06-04 09:32:00.000.

```
SET dtm_val TO make_datetime$(200, 572)
```

**See also** **date\_to\_mdy\$( )**, **mdy\_to\_date\$( )**, **null\_convert\$( )**, **str\_to\_date\$( )**, **to\_amount\$( )**, **to\_binary\$( )**, **to\_bool\$( )**, **to\_date\$( )**, **to\_datetime\$( )**, **to\_float\$( )**, **to\_num\$( )**, **to\_string\$( )**, **to\_text\$( )**, **to\_time\$( )**, **val\_to\_str\$( )**

## current\_datetime\$( )

*System function*

*User environment*

**Syntax** **current\_datetime\$( )**

**Return Values** (**DATETIME**) The **DATETIME** value of the current date and time as retrieved from the operating system.

**Description** The **current\_datetime\$( )** system function obtains the current date and time from the operating system and returns them as an ACCELL/SQL **DATETIME** value. The precision of the value returned depends on the precision of the operating system's *gettimeofday( )* call and may be less than the full microsecond precision of an ACCELL/SQL **DATETIME**.

**Example** In the following example, **current\_datetime\$( )** assigns the current date and time to the **dtm\_entered** variable if **dtm\_entered** is undefined:

```
FIELD dtm_entered
  AFTER FIELD
    IF (dtm_entered IS UNDEFINED ) THEN
      SET dtm_entered TO current_datetime$( )
```

**See also** **current\_date\$( )**, **current\_time\$( )**

## datetime\_to\_mdym\$( )

*System function*

*Datetime conversion*

**Syntax** **datetime\_to\_mdym\$( datetime, month, day, year, microseconds )**

**Arguments** *datetime* (**DATETIME**) The **DATETIME** value to be converted.

*month* (**NUMERIC**) A variable to receive the month of the converted **DATETIME**. This parameter is an ACCELL/SQL result parameter.

- day* (NUMERIC) A variable to receive the day of the converted **DATETIME**. This parameter is an ACCELL/SQL result parameter.
- year* (NUMERIC) A variable to receive the year of the converted **DATETIME**. This parameter is an ACCELL/SQL result parameter.
- microseconds* (NUMERIC) A variable to receive the number of microseconds since midnight of the converted **DATETIME**. This parameter is an ACCELL/SQL result parameter.

*Return Values* None.

*Description* The **datetime\_to\_mdym\$( )** system function converts a **DATETIME** value to four **NUMERIC** values: *month*, *day*, *year* and *microseconds*. These four values correspond to the month, day and year represented by the **DATETIME** value, and the number of microseconds since midnight of the date represented by the value. The conversion is done in the time zone of the current session. Due to the large numeric values that may be returned in the *microseconds* argument by this function, the **UNUMERIC64** configuration variable should be set to **TRUE** to allow ACCELL/SQL to support 64-bit integer values. Failure to do so will result in numeric overflow errors. If *datetime* is **NULL**, all of *month*, *day*, *year*, and *microseconds* will be set to **NULL**. This function is the inverse of **mdym\$( )**.

*See also* **mdym\$( )**, **mdy\_to\_date\$( )**, **date\_to\_mdy\$( )**

## mdym\$( )

*System function*

*Datetime conversion*

*Syntax* **mdym\$( month, day, year, microseconds )**

### *Arguments*

- month* (NUMERIC) A variable containing the month to be converted to the **DATETIME** value. Must be an integer in the range 1 to 12.
- day* (NUMERIC) A variable containing the day to be converted to the **DATETIME** value. Must be an integer in the range 1 to 31.
- year* (NUMERIC) A variable containing the year to be converted to the **DATETIME** value. Must be given as a four digit number between 1752 and 9999. For example, you would use **1988** (not **88**) for the year 1988.
- microseconds* (NUMERIC) A variable containing the microseconds since midnight to be converted to the **DATETIME** value. Must be in the range 0 to 8639999999.

*Return Values* (**DATETIME**) The **DATETIME** value represented by the input values.

*Description* The **mdym\$( )** system function converts four numeric values: *month*, *day*, *year* and *microseconds* to a **DATETIME** value. The conversion is done in the time zone of the current session. This function is the inverse of **datetime\_to\_mdym\$( )**. If any of the values *month*, *day*, *year* or *microseconds* is **NULL**, the return value will be **NULL**.

See also `datetime_to_mdym$( )`, `mdy_to_date$( )`, `date_to_mdy$( )`

---

## Changed system functions for **DATETIME** support

---

### `to_num$( )`

The `to_num$( )` system function has been enhanced to support **DATETIME** values. If the argument is a **DATETIME**, the inverse of `to_datetime$( )` is performed and the value returned is the number of microseconds **since January 1st, 1970 00:00:00.000000 GMT** (the epoch). Negative values will be returned for dates and times prior to the epoch. Due to the large numeric values that may be returned by this function when called to convert a **DATETIME** value, the **UNUMERIC64** configuration variable should be set to **TRUE** to allow ACCELL/SQL to support 64-bit integer values. Failure to do so will result in numeric overflow errors.

---

### `to_string$( )`

The `to_string$( )` system function has been enhanced to support **DATETIME** values. If the argument is a **DATETIME** it will be converted to a string representation using the format represented by the **UDATETIMEFMT** and **DATETIMENULLCH** configuration variables.

---

### `to_string_using$( )`

The `to_string_using$( )` system function has been enhanced to support **DATETIME** values. If the argument is a **DATETIME** it will be converted to a string representation using the format specified. See the section *Input and output formatting of DATETIME values* on page 6 for a description of the format specification.

---

## Using **DATETIME** in RPT

The **DATETIME** data type is available for use in **RPT**.

---

### Reading **DATETIME** values through the **INPUT** section

Input variables may be declared in the **input** section of the **RPT** script as type **datetime**. The input values will be interpreted according to the setting of the **UDATETIMEFMT** configuration variable.

---

### Printing **DATETIME** values

**DATETIME** variables in an **RPT** script may be used in the **print** statement. If no **using** clause is specified, the variable will be formatted according to the setting of the **UDATETIMEFMT** configuration variable. The **using** clause for a **DATETIME** variable can be specified using the same syntax as the **UDATETIMEFMT** configuration variable (described above in the section *Input and output formatting of DATETIME values*.)

---

## Apache Tomcat servlet/JSP engine installed for use with Accell/Web and RPT/Web

In place of the Unify eWave Engine (ACCELL/SQL 8.2 and earlier) or JBoss (ACCELL/SQL 8.3-9.0) application servers, release 9.1 of ACCELL/SQL includes Apache Tomcat 7 as the default servlet/JSP engine provided for use with Accell/Web and RPT/Web. Following are simplified instructions for using Tomcat with ACCELL/SQL. For detailed Apache Tomcat documentation, please visit <http://tomcat.apache.org/tomcat-7.0-doc/index.html>.

---

### Installation

The included version of Tomcat will be installed as part of running *install/install* during ACCELL/SQL installation. When configuring Tomcat, the installation process will first request the location of the web application deploy directory. This is a directory where web applications (RPT/Web and Accell/Web applications) should be deployed. The default location for the deploy directory is */opt/tomcatdeploy*.

There are also three network ports that must be configured:

- Shutdown Port (default 8005) – The port used by the *shutdown.sh* script to safely shut down Tomcat.
- HTTP Port (default 8080) – The port on which web applications will be available to browsers.
- AJP Port (default 8009) – The port used to proxy web applications through another HTTP server, such as Apache HTTP Server.

The installer allows the option to automatically configure the ports to the above defaults. If the default ports cannot be used (for example if one of the ports is in use by another service) the ports can be configured manually.

---

### Starting Tomcat

To start up Tomcat, first set the environment variable **JAVA\_HOME** to point to the location of a Java 6 JRE or JDK. A compatible JDK is included in the Accell release in the *j2sdk* directory (*\$UNIFY/./j2sdk*). Once **JAVA\_HOME** is set, start Tomcat by executing the script *bin/startup.sh* in the Tomcat release directory (*\$UNIFY/./tomcat*). You should see output similar to the following:

```
$ JAVA_HOME=$UNIFY/./j2sdk
$ export JAVA_HOME
$ $UNIFY/./tomcat/bin/startup.sh
Using CATALINA_BASE:   /work/accell/ac191/release/tomcat
Using CATALINA_HOME:   /work/accell/ac191/release/tomcat
Using CATALINA_TMPDIR: /work/accell/ac191/release/tomcat/temp
Using JRE_HOME:        /work/accell/ac191/release/lib/./j2sdk
Using CLASSPATH:
/work/accell/ac191/release/tomcat/bin/bootstrap.jar:/work/accell/ac191/release/tomcat/bin
/tomcat-juli.jar
$
```

At this point any deployed web applications should be visible in a browser using URLs beginning with

```
http://localhost:http_port/
```

where *http\_port* is the HTTP port specified during installation. For example:

```
http://localhost:8080/reports
```

should display the RPT/Web repositories.

---

## Stopping Tomcat

To gracefully stop the Tomcat server, set the **JAVA\_HOME** environment variable as described above, and run the *bin/shutdown.sh* script in the Tomcat release directory. For example:

```
$ JAVA_HOME=$UNIFY/../j2sdk
$ export JAVA_HOME
$ $UNIFY/../tomcat/bin/shutdown.sh
Using CATALINA_BASE:   /work/accell/ac191/release/tomcat
Using CATALINA_HOME:   /work/accell/ac191/release/tomcat
Using CATALINA_TMPDIR: /work/accell/ac191/release/tomcat/temp
Using JRE_HOME:        /work/accell/ac191/release/lib/../j2sdk
Using CLASSPATH:
/work/accell/ac191/release/tomcat/bin/bootstrap.jar:/work/accell/ac191/release/tomcat/bin
/tomcat-juli.jar
$
```

---

## Deploying/Undeploying web applications

To deploy a web application, simply copy it to the deploy directory (for example */opt/tomcatDeploy*) specified during installation. You can deploy applications either contained in a WAR file or as a directory containing the application. The URL context of the application will be the name of the WAR file (minus the *.war* extension) or the name of the directory containing the application. For example, the RPT/WEB application is contained in a WAR named *reports.war* which is copied to the deploy directory during installation. In a default installation, the URL to access the RPT/Web application would be

```
http://localhost:8080/reports
```

An Accell/Web application deployed as a directory named “MyApp” would be accessed through the URL

```
http://localhost:8080/MyApp
```

To undeploy an application, simply delete the WAR file or the application directory from the deploy directory.

---

## RPTDIVIDEBYZERO configuration variable

The RPT utility in ACCELL/SQL and DataServer releases prior to 8.1 did not complain when dividing by zero, but rather produced a zero result. This behavior changed in 8.1 as a side effect of new development. Division by zero began to result in a string of question marks (?????); an undefined value. Since reports written before 8.1 may fail due to that change, the previous behavior was restored, and configuration variable was created for reports which require the new behavior.

The new configuration variable also supports a compromise behavior. Use it for reports which depend on division by zero resulting in zero when the numerator is zero, but which must fail for division by zero calculations where the numerator is non-zero.

New configuration variable:

### **RPTDIVIDEBYZERO**

Use this variable to specify division by zero behavior for RPT.

Valid values:

- **INDETERMINATE** - Division by zero is undefined, regardless of the value of the numerator.
- **ZIFZNUMERATOR** - Zero will be the result if the numerator is zero, otherwise the result is undefined.
- **ALWAYSZERO** - The result of division by zero is always zero.

Default value: **ALWAYSZERO**

Example: RPTDIVIDEBYZERO="ZIFZNUMERATOR"

If RPTDIVIDEBYZERO is set to an invalid string, the default will be assumed.

---

## New configuration variable **ACL\_LIC\_WARN**

A new configuration variable **ACL\_LIC\_WARN** has been added to ACCELL/SQL that specifies whether ACCELL/SQL and ACCELL/Web will display warnings and allow use when the number of runtime licenses is exceeded.

Valid values:

- **TRUE:** Display warning messages when all of the runtime licenses are in use, and fail when the count has grown to about 10% more than the allowed limit (the behavior prior to introducing this variable).
- **FALSE:** Do not display warnings, and fail immediately when all of the runtime licenses are in use.

The default value for the variable is **TRUE**.