Orchestrate 7.0.1 Release Notes

These release notes contain important information about the new and enhanced features added to Orchestrate 7.0.1. These notes are a supplement to the documentation.

Decimal Field Changes

New Decimal Rounding Mode

Orchestrate now rounds towards the nearest representable value. For *x*`.5` values, it rounds towards positive infinity; for `-`*x*`.5` values, it rounds toward negative infinity. This mode corresponds to the `COBOL ROUNDED` function.

Examples are:

`1.4` is rounded to `1`
`1.5` is rounded to `2`
`-1.4` is rounded to `-1`
`-1.5` is rounded to `-2`

Previously, Orchestrate simply discarded the decimal fraction, which is a rounding mode that corresponds to the `COBOL INTEGER-PART` function. Examples are:

`1.9` was rounded to `1`
`-8.1` was rounded to `-8`

Fatal errors may occur at runtime if the precision of the destination decimal is smaller than that of the source decimal.

Using Decimals in the group Operator

With the **-countType** suboption, you can now specify that the **group** operator **-reduce** option fields and the **-rereduce** option summary fields be handled as **decimal** fields instead of, by default, **dfloat** fields.

Use this syntax:

`-reduce` *field_name* `[-countType Double | Decimal]`

`-rereduce` *source_field* `-summary-field` *summary_field*
`[-countType Double | Decimal]`

Addition and subtraction operations are more efficient with **decimal** fields; multiplication and division operations are more efficient with **dfloat** fields.

New Functions for the APT_Decimal Class

Arithmetic Functions

- `APT_Status addDecimal(const APT_Decimal& d1, const APT_Decimal& d2, int scaleAdj = APT_DEFAULT_SCALE_VALUE, RoundMode r=eRoundInf, bool checkDecimal=false);`
- `APT_Status subtractDecimal(const APT_Decimal& d1, const APT_Decimal& d2, int scaleAdj = APT_DEFAULT_SCALE_VALUE, RoundMode r=eRoundInf, bool checkDecimal=false);`
- `APT_Status multiplyDecimal(const APT_Decimal& d1, const APT_Decimal& d2, int scaleAdj = APT_DEFAULT_SCALE_VALUE, RoundMode r=eRoundInf, bool checkDecimal=false);`
- `APT_Status divideDecimal(const APT_Decimal& d1, const APT_Decimal& d2, int = APT_DEFAULT_SCALE_VALUE, RoundMode r=eRoundInf, bool checkDecimal=false);`

The `scaleAdj` argument defines what should be done with the scale and precision of the result. There are two possibilities:

- If the `scaleAdj` is positive, the scale of the result is equal to `scaleAdj`; but it may be diminished to keep the precision not greater than 255.

  or

- `scaleAdj` may have one of these special values:
  - `APT_ACCEPT_DEFAULT_VALUE -4`
  - `APT_LEAVE_SCALE_UNCHANGED -3`
  - `APT_LEAVE_ONLY_TRUSTED_DIGITS -2`

- APT_KEEP_ALL_DIGITS -1

Neither the precision nor scale of record-based fields may be changed.

Overridden Functions

- **operator+( )**
- **operator-( )**
- **operator*( )**
- **operator/( )**

For more information, refer to `$APT_ORCHHOME/include/apt_util/decimal.h`, the header file for the **APT_Decimal** class.

New Environment Variables

- `APT_DECIMAL_INTERM_PRECISION` *precision*
  `APT_DECIMAL_INTERM_SCALE` *scale*
  `APT_DECIMAL_INTERM_ROUNDMODE` *round_mode*

  These variables define the precision and scale for the intermediate decimals created by the decimal operators `+`, `-`, `*`, and `/`, and the round mode used by these operators. When these variables are not set, the default values are given by the **DEFAULT_INTERM_PRECISION** and **DEFAULT_INTERM_SCALE** define directives and by **APT_Decimal::eRoundInf.**

- `APT_SCALE_VALUE_FOR_SUM` *value*

  If the three functions listed below are called with `scale = APT_ACCEPT_DEFAULT_VALUE`, this variable defines the scale and precision adjustments for the results.

  - **APT_Decimal::addDecimal( )**
  - **APT_Decimal::subtractDecimal( )**
  - **APT_Decimal::addOrSubtract( )**

- `APT_SCALE_VALUE_FOR_PRODUCT`

  If the three functions listed below are called with `scale = APT_ACCEPT_DEFAULT_VALUE`, this variable defines the scale and precision adjustments for the results.

  - **APT_Decimal::multiplyDecimal( )**
  - **APT_Decimal::divideDecimal( )**
  - **APT_Decimal::multiplyOrDivide( )**

- `APT_CHECK_DECIMAL_VALIDITY`

  Requires that the overridden `operator=(const APT_Decimal& src)` functions as `assignFromDecimal(const APT_Decimal& src)` does; that is, checks the validity of `src`. By default, the validity of src is not checked, optimizing performance significantly.

See `$APT_ORCHHOME/include/apt_util/decimal.h` for more information.

New Options for the db2load Operator

-nonrecoverable Option

This option prevents a database backup-pending state after **dbload** has finished execution. Its syntax is:

```
[-nonrecoverable]
```

-exceptionTable Option

This option specifies a table for inserting records which violate load-table constraints. Its syntax is:

`[-exceptionTable` *table_name*`]`

The exceptions table should be consistent with the `FOR EXCEPTION` option of the **db2load** utility. Refer to the `DB2` documentation for instructions on how to create this table.

The **-exceptionTable** option cannot be used with the **create** or **replace** modes because Orchestrate cannot recreate the table with all its applicable data constraints.

When the mode is **truncate**, the tables for the **-table** and **-exceptionTable** options are truncated.

-statistics Option

This option specifies which statistics should be generated upon load completion. This option is only valid with the **truncate** mode; it is ignored for other modes. Its syntax is:

`[-statistics` *value*`]`

where *value* can be one of these values:

```
stats_none
stats_exttable_only
stats_extindex_only
stats_exttable_index
stats_index
stats_table
```

```
stats_extindex_table
stats_all
stats_both
```

The default value is `stats_none`.

As a part of the loading process `DB2` collects the requisite statistics for table-access optimization. Alternatively, you can run the `RUNSTAT` utility; refer to Chapter 5 of the Administration Guide, Vol. 3.

-cpu Option

This option specifies the number of processes to initiate on every node. Its syntax is:

```
[-cpu integer]
   [-anyorder]
```

The default value for the **-cpu** option is `1`. Specifying a `0` value allows **db2load** to generate an algorithm that determines the optimal number based on the number of CPUs available at runtime. Note that the resulting value may not be optimal because `DB2` does not take into account the Orchestrate workload.

The **-anyorder** suboption allows the order of loading to be arbitrary for every node, potentially leading to a performance gain. The **-anyorder** option is ignored if the value of the **-cpu** option is `1`.

Arithmetic Extensions to the transform Operator

The Transformation Language has been extended to support the placement of binary operators between two decimals and between a decimal and one of the numeric field types.

The binary operators are `+`, `-`, `*`, and `/`.

The numeric field types are **int8**, **uint8**, **int16**, **uint16**, **int32**, **uint32**, **int64**, **uint64**, **sfloat**, and **dfloat**.

Generally there are no restrictions on the form of an operand. It can be a constant, a variable, a function call that returns a numeric value, a simple expression such as an addition of two variables, or a complicated expression that consists of function calls as well as arithmetic operations.

By default, the **transform** operator sets the precision and scale of any temporary internal decimal variable created during arithmetic operations to [`TRX_DEFAULT_MAX_PRECISION=38,TRX_DEFAULT_MAX_SCALE=10`] with **RoundMode** equal to **eRoundInf**.

You can override the default values using these environment variables:

- `APT_DECIMAL_INTERM_PRECISION` *value*
- `APT_DECIMAL_INTERM_SCALE` *value*
- `APT_DECIMAL_INTERM_ROUNDMODE ceil | floor | round_inf | trunc_zero`

Fatal errors may occur at runtime if the precision of the destination decimal is smaller than that of the source decimal.

Enhancement to the generator Operator cycle Option

You can now use a simple expression for the **init** component of the **cycle** option when the **part** keyword or the **partcount** keyword is used.

The syntax for the **cycle** option is now:

```
cycle = {init = 'part | partcount operator constant',
         incr = incr_val, limit = limit_val }
```

where *operator* can be `+`, `-`, `*`, or `/`.

For example:

```
generator -schema record(a:int64 {cycle = {init = 'part + 100',
                                           incr = 1, limit = 1000}};)
```

Establishing a Remote Connection to the hplread Operator

This section describes how to set up the **hplread** operator to run on a remote machine without having INFORMIX installed on your local machine.

- **To establish a remote connection to the hplread operator:**

  1. Verify that the INFORMIX `sqlhosts` file on the remote machine has a TCP interface. A TCP interface is necessary to use the remote connection functionality.
  2. Copy the INFORMIX `etc/sqlhosts` file from the remote machine to a directory on your local machine. Set the INFORMIX `INFORMIXDIR` environment variable to this directory.

     For example, if the directory on the local machine is `/apt/informix`, the `sqlhosts` file should be in the directory `/apt/informix/etc`, and the `INFORMIXDIR` variable should be set to `/apt/informix`.
  3. Set the `INFORMIXSERVER` environment variable to the name of the remote INFORMIX server.

4 Add the remote INFORMIX server nodes to your node configuration file located in `$APT_ORCHHOME/../../config`; and use a nodepool resource constraint to limit the execution of the **hplread** operator to these nodes.

In the example configuration file below, the local machine is `fastname local_machine`, and the INFORMIX remote server machine is `fastname remote_machine`. The nodepool for the remote nodes is arbitrarily named `"InformixServer"`. The configuration file must contain at least two nodes, one for the local machine and one for the remote machine.

Here is the example configuration file before any changes have been made:

```
{
node "node0"
   {
   fastname "local_machine"
   pools "" "node0" "local_machine"
   resource disk "/orch/s0" {}
   resource disk "/orch/s1" {}
   resource scratchdisk "/scratch" {}
   }
node "node1"
   {
   fastname "local_machine"
   pools "" "node1" "local_machine"
   resource disk "/orch/s0" {}
   resource disk "/orch/s1" {}
   resource scratchdisk "/scratch" {}
   }
}
```

Here is the example configuration file with changes made for the **hplread** operator:

```
{
node "node0"
   {
   fastname "local_machine"
   pools "" "local_machine"
   resource disk "/orch/s0" {}
   resource disk "/orch/s1" {}
   resource scratchdisk "/scratch" {}
   }
node "node1"
   {
   fastname "local_machine"
   pools "" "local_machine"
   resource disk "/orch/s0" {}
   resource disk "/orch/s1" {}
   resource scratchdisk "/scratch" {}
   }
node "node2"
   {
   fastname "remote_machine"
   pools "InformixServer" "remote_machine"
```

```
    resource disk "/orch/s0" {}
    resource disk "/orch/s1" {}
    resource scratchdisk "/scratch" {}
    }
node "node3"
    {
    fastname "remote_machine"
    pools "InformixServer" "remote_machine"
    resource disk "/orch/s0" {}
    resource disk "/orch/s1" {}
    resource scratchdisk "/scratch" {}
    }
}
```

5. Remote access to an INFORMIX database requires the use of two `INFORMIXDIR` environment variable settings, one for the local machine which is set as in **Step 2** above, and one for the machine with the remote INFORMIX database. The remote variable needs to be set in a startup script which you must create on the local machine. This startup script is executed automatically by Orchestrate.

   Here is a sample `startup.apt` file with `INFORMIXDIR` being set to `/usr/informix/9.4`, the INFORMIX directory on the remote machine:

   ```
   #! /bin/sh
   INFORMIXDIR=/usr/informix/9.4
   export INFORMIXDIR
   shift 2
   exec $*
   ```

6. Set the environment variable `APT_STARTUP_SCRIPT` to the full pathname of the `startup.apt` file.

7. You are now ready to run a job which uses the **hplread** operator to connect to a remote INFORMIX server. If you are unable to connect to the remote server, try making either one or both of the following changes to your `sqlhosts` file on the local machine:

   - In the fourth column in the row corresponding to the remote INFORMIX server name, replace the INFORMIX server name with the INFORMIX server port number found in the `/etc/services` file on the remote machine.
   - The third column contains the hostname of the remote machine. Change this to the IP address of the remote machine.

lookup Operator Table Incompatibility

The **lookup** operator has been redesigned to increase its performance. This process has altered the structure of lookup tables, making the tables generated prior to Orchestrate 7.0.1 incompatible with this release. Consequently, it is necessary to generate new lookup tables.

You use the **-createOnly** option of the **lookup** operator to generate a lookup table. See the *lookup Operator* chapter of the *Orchestrate 7.0 Operators Reference* for the details.

Forcing a Fixed Read/Write Size

Setting the new `APT_CONSISTENT_BUFFERIO_SIZE` environment variable to a value equal to the read/write size in bytes forces a fixed read/write size on IO during import and export. This can be useful with certain disk arrays to optimize performance.

Rejecting Records with String Fields that Exceed the Maximum Length

You can now use the **APT_IMPORT_REJECT_STRING_FIELD_OVERRUNS** environment variable to direct Orchestrate to transfer records with strings longer than their declared maximum length to the reject data set.

By default, imported string fields that exceed their maximum declared length are automatically truncated.

Loading Delimited Files with the orawrite Operator

You can now specify a field delimiter character for the **orawrite** operator using the new **APT_ORACLE_LOAD_DELIMITED** environment variable. Setting this variable makes it possible for **orawrite** to load fields with trailing or leading blank characters. When this variable is set, but does not have a value, the comma (,) character is used as the default delimiter.

Using the null_field Property when Importing

When you are specifying the **null_field** property at the record level, it is important to also specify the field-level **null_field** property to any nullable fields not covered by the record-level property.

For example:

```
record { null_field = 'aaaa' }
   ( field1:nullable int8 { null_field = '-127' };
     field2:nullable string[4];
     field3:nullable string;
```

```
  field4:nullable ustring[8] {null_field = 'ââââââââ' };
)
```

The record-level property above applies only to variable-length strings and fixed-length strings of four characters, `field2` and `field3`; `field1` and `field4` are given field-level **null_field** properties because they are not covered by the record property.

Enabling copy Operator Insertion

Set the **APT_INSERT_COPY_BEFORE_MODIFY** environment variable to enable the automatic insertion of a **copy** operator before a **modify** operator. This process ensures that your data flow does not have contiguous **modify** operators, a practice which is not supported in Orchestrate.

When this variable is not set and the operator immediately preceding a **modify** operator in the data flow also includes a **modify** operator, Orchestrate removes the downstream **modify** operator.

SAS Resources

Specifying the Location of Your SAS Executable

To locate your SAS executable, Orchestrate uses the following sources in this order:

1. The absolute path you specify to the `APT_SAS_COMMAND` environment variable or the `APT_SASINT_COMMAND` environment variable.

   You use `APT_SAS_COMMAND` to specify the location of a basic US SAS executable. For example:

   ```
   APT_SAS_COMMAND /usr/local/sas/sas8.2/sas
   ```

   You use `APT_SASINT_COMMAND` to specify the location of a SAS International executable. The path includes `/dbcs`. For example:

   ```
   APT_SASINT_COMMAND /usr/local/sas/sas8.2int/dbcs/sas
   ```

2. The `resource sas` specification in your configuration file. Use this syntax:

   ```
   resource sas "absolute_path" { }
   ```

   For example:

   ```
   resource sas /usr/local/sas/sas8.2 { }
   ```

3. The path to SAS in your `$PATH` environment variable. Do not include a trailing `sas` in this path. Examples:

   `/usr/local/sas/sas8.2` (for basic US SAS)

   `/usr/local/sas/sas8.2int/dbcs` (for SAS International)

Determining Which SAS Version is Accessed

You can run the `sanitytest` script to determine what version of SAS is accessed:

```
osh -f $APT_ORCHHOME/examples/sas/sanitytest
```

Your SAS system is capable of running in international mode if your SAS log output has this type of header:

```
NOTE: SAS (r) Proprietary Software Release 8.2 (TS2MO DBCS2944)
```

When you have invoked SAS in standard mode, your SAS log output has this type of header:

```
NOTE: SAS (r) Proprietary Software Release 8.2 (TS2MO)
```

Specifying a Character Set for ustring Values

You can specify what character set Orchestrate uses to map between your **ustring** values and the `char` data stored in SAS files. You use the **-sas_cs** option of the SAS-interface operators to indicate your character-set choice.

The syntax for the **-sas_cs** option is:

`-sas_cs` *icu_character_set* | *DBCSLANG*

In `$APT_ORCHHOME/etc/`*platform*`/` there are platform-specific `sascs.txt` files that list each DBCSLANG setting. The *platform* directory names are: `sun`, `aix`, `osf1` (`Tru64`), `hpux` , and `linux`. For example:

```
$APT_ORCHHOME/etc/aix/sascs.txt
```

When you specify an ICU character setting, the `sascs.txt` file must be located in the platform-specific directory for your operating system. By mapping between `DBCSLANG` and `ICU` settings, Orchestrate accesses the setting that is equivalent to your **-sas_cs** specification for your operating system. `ICU` settings can differ between platforms.

For each `DBCSLANG` setting you use, enter your `ICU` equivalent. For example, in the following `sascs.txt` mapping table, `ISO-2022-JP` is entered for the `JAPANESE DBCSLANG` setting.

DBCSLANG Setting	ICU Character Set
`JAPANESE`	`ISO-2022-JP`
`KATAKANA`	*icu_character_set*
`KOREAN`	*icu_character_set*
`HANGLE`	*icu_character_set*
`CHINESE`	*icu_character_set*
`TAIWANESE`	*icu_character_set*

string_trim Function for the modify Operator

The syntax of the string_trim function is:

stringField=`string_trim`[*character, direction, justify*] (*string*)

You can use this function to remove the characters used to pad variable-length strings when they are converted to fixed-length strings of greater length. By default, these characters are retained when the fixed-length string is then converted back to a variable-length string.

The *character* argument is the character to remove. It defaults to `NULL`. The value of the *direction* and *justify* arguments can be either `begin` or `end`; *direction* defaults to `end`, and *justify* defaults to `begin`. *justify* has no affect when the target string has variable length.

This example removes all leading ASCII NULL characters from the beginning of name and places the remaining characters in an output variable-length string with the same name:

```
name:string = string_trim[NULL, begin](name)
```

This example removes all trailing `Z` characters from `color`, and left-justifies the resulting `hue` fixed-length string:

```
hue:string[10] = string_trim['Z', end, begin](color)
```

Teradata v2r5 Support

The Teradata operators now support Teradata v2r5.

Performance Improvement for Bounded Variable-Length Fields

Bounded variable-length field declarations include the **max** field property that specifies the maximum number of code points a field can contain. The efficiency of processing these fields has been optimized in this release.

final_delim import/export Property Change

By default, on export a space is now inserted after every field except the last field in the record. Previous to this release, a space was inserted after every field, including the last field; or, when **delim** property was set, its value was used instead of the **final_delim** value.

Now when you specify the **final_delim** property for records that have a tagged or subrec field as the last field, your specification is correctly applied unless the subrec is a vector. By default, a space is added to the last field when it is a subrec vector.

You can set the `APT_PREVIOUS_FINAL_DELIM_COMPATIBLE` environment variable to obtain the **final_delim** behavior prior to this release.

The Orchestrate Documentation Set

The following pdf documents are in your `$APT_ORCHHOME/doc/pdf/` directory. The *7.0.1BookShelf* file provides links to these documents.

- *Orchestrate 7.0 User Guide*
- *Orchestrate 7.0 Operators Reference*
- *Orchestrate 7.0 Installation and Administration Manual*
- *Orchestrate 7.0 Developer Guide*
- *Orchestrate 7.0 WebHouse User Guide*
- *Orchestrate 7.0 Record Schema*
- *Orchestrate 7.0 C++ Classes and Macros Sorted by Header File*
- *Orchestrate 7.0 C++ Classes and Macros Sorted by Name*

Searching for Text in Orchestrate Documents

You can find specific words or phrases in an Orchestrate online document and across all Orchestrate online documents using the Adobe 6.0 Acrobat Reader. The Reader is available as a free download from adobe.com.

Assistance and Additional Information

If you require assistance or have questions about Orchestrate, you can contact Ascential Customer Support by:

- Calling (866) INFONOW in North America or your Regional Support Center.
- Writing support@ascential.com for any Ascential Software product or orch-support@ascential.com for Orchestrate-specific help.
- Logging onto the Ascential Support e.Service Web site at:

  www.ascential.com:8080/eservice/index.html

For current information about Ascential and its products log onto:

http://www.ascential.com/

Legal Notices

Copyright, trademark, and open source legal notices pertaining to Orchestrate are listed in `$APT_ORCHHOME/doc/pdf/LegalNotices.pdf`.

Supported Operating Systems and Applications

Operating Systems	C++ Compiler	DBMS	SAS
AIX 4.3.3, 5.1, and 5.2	VisualAge C++ 5.0.2.0 and 6.0	IBM DB2 UDB 7.2 EEE and DB2 UDB ESE v8.1 with DPF INFORMIX XPS 8.3 and 8.4 Oracle 8i EE R3 (8.1.7) for AIX 4.3.3 and 5.1 only; Oracle 9i (9.2) for AIX 4.33, 5.1, and 5.2 Teradata v2r4.1 and v2r5 Teradata Utilities Foundation (TUF) 6.1.0 and TTU 7.0	SAS 6.12 and 8.2
Compaq Tru64 5.1	Compaq C++ 6.2, 6.3, and 6.5	Oracle EE R3 (8.1.7) and Oracle 9i (9.2)	SAS 6.12 and 8.2
Linux Redhat Red Hat Enterprise Linux AS 2.1	gcc/g++ 2.96	Oracle 8i EE R3 (8.1.7) and Oracle 9i (9.2) DB2 UDB ESE v8.1 with DPF	SAS 8.2
HP-UX 11 and 11.11	HP ANSI C++ A3.33 and A3.37 **Note**: The A3.33 compiler requires patch PHSS_29483 from HP. Contact Ascential Client Support for information.	IBM DB2 UDB 7.2 EEE and DB2 UDB ESE v8.1 with DPF INFORMIX XPS 8.3 and 8.4 Oracle 8i EE R3 (8.1.7) for HP-UX 11.1 only; Oracle 9i (9.2) for 11 and 11.11 Teradata v2r4.1 and v2r5 Teradata Utilities Foundation (TUF) 6.1.1 and TTU 7.0	SAS 6.12 and 8.2
Sun Solaris for Sparc 2.7, 2.8, and 2.9.	Sun Pro C++ 6.0/ Forte Sun One Studio 7	IBM DB2 UDB 7.2 EEE and DB2 UDB ESE v8.1.2003 with DPF INFORMIX XPS 8.3 and 8.4 Oracle EE R3 (8.1.7) for Solaris 2.9 only; 9i (9.2) for Solaris 2.7, 2.8, and 2.9 Teradata v2r4.1 and v2r5 Teradata Utilities Foundation (TUF) 6.1.1 and TTU 7.0	SAS 6.12 and 8.2