Interaction Properties Reference Manual

Translations

4/5/2026

Contents

Translations

In IRD, on the `Conditions` and `Order` tabs of a View object in an Business Process, you can write statements that refer to the three types of interaction properties that are represented as independent fields in the`interactions` table: system, business, and custom. You can also use built-in translations, or functions, which provide database manipulation tools that are independent of the underlying database.

Translations enable you to use the interaction property names as presented in this chapter rather than database table field names. They provide a unified representation, regardless of the underlying database, of a collection of functions. Some deal with time and date, others with attributes of the specific interaction; for example, `_age()` calculates the interaction's age in seconds, regardless of database type used.

> **Important**
>
> Bear in mind the following:
>
> - Translation does not hide the original database syntax. Interaction Server translates whatever it can, leaves the rest unchanged, and transmits it all to the database. Therefore you can also use database field names and any database-specific constructs on the `Conditions` and `Order` tabs.
> - Many of these translations are time sensitive. As such they have a dependency on the `freeze-interval` option of the Interaction Queue View object. For details, see the discussion of this option in the "Interaction Server Options" section of the "Configuration Options" chapter of the *eServices 8.1 Reference Manual.*

_timestamp

Specifies date constants in conditions. The single argument is a character constant that represents a date in the common form used by Interaction Server.

Usage

```
_timestamp('yyyy-mm-ddThh:mi:ssZ')
```

OR

```
_timestamp('yyyy-mm-dd hh:mi:ss')
```

Translations Performed by Database

For Oracle 9 and 10:

```
 TO_DATE('yyyy-mm-dd hh:mi:ss', 'YYYY-MM-DD HH24:MI:SS')
```

For Microsoft SQL:

```
CONVERT(DATETIME, 'yyyy-mm-ddThh:mi:ssZ', 102)
```

For DB2:

```
TIMESTAMP('yyyy-mm-dd hh:mm:ss')
```

_current_time

Calculates the current UTC date-time.

Use this function to avoid the confusion caused by mixing UTC and local times. Since Interaction Server manipulates only UTC time, and all dates in the database are saved as UTC time, there is no standard function to get the current local date-time.
Database-specific functions can also be used to get the local time, such as Microsoft SQL's `getdate()` or Oracle's `sysdate`.

Usage

```
_current_time()
```

Translations Performed by Database

For Oracle 9 and 10:

```
cast(SYS_EXTRACT_UTC(SYSTIMESTAMP) as date)
```

For Microsoft SQL:

```
GETUTCDATE()
```

For DB2:

```
(CURRENT TIMESTAMP - CURRENT TIMEZONE)
```

_timestampdiff

Calculates the difference, in seconds, between two timestamps.

Both arguments should be of type timestamp, which differs depending on the database engine:

- Oracle—date
- Microsoft SQL—datetime
- DB2—timestamp

Usage

```
_timestampdiff(timestamp date1, timestamp date2)
```

Translations Performed by Database

For Oracle 9 and 10:

```
 (((date1)</tt> - <tt>(date2))*86400)
```

For Microsoft SQL:

```
cast(cast(((date1) - (date2)) as float)*86400 as integer)
```

For DB2:

```
 timestampdiff( 2, char((date1)</tt> - <tt>(date2)) )
```

_timestampadd

Method of increasing and decreasing dates. Generates a date increased or decreased by a specified number of seconds. The second parameter can be negative, to make the result a date earlier than the original date.

Usage

```
_timestampadd(timestamp date, integer seconds)
```

Translations Performed by Database

For Oracle 9 and 10:

```
 (date + (seconds)/86400)
```

For Microsoft SQL:

```
 dateadd(second, seconds, date)
```

For DB2:

```
 (date + (seconds) second)
```

_age

Calculates the age in seconds of the interaction; that is, the difference between the current time in UTC and the interaction attribute `ReceivedAt` .

Usage

```
_age()
```

Translations Performed by Database

For Oracle 9 and 10:

```
 ((cast(SYS_EXTRACT_UTC(SYSTIMESTAMP) as date)</tt> - <tt>received_at)*86400)
```

For Microsoft SQL:

```
cast(cast((getutcdate() - received_at) as float)*86400 as integer)
```

For DB2:

```
timestampdiff( 2, char((current timestamp - current timezone)</tt> - <tt>received_at))
```

_time_in_queue

Calculates the time, in seconds, that the interaction has spent in the queue; that is, the difference between the current time in UTC and the attribute `PlacedInQueueAt`.

Usage

```
_time_in_queue()
```

Translations Performed by Database

For Oracle 9 and 10:

```
((cast(SYS_EXTRACT_UTC(SYSTIMESTAMP) as date)</tt> - <tt>placed_in_queue_at)*86400)
```

For Microsoft SQL:

```
cast(cast((getutcdate()</tt> - <tt>placed_in_queue_at) as float)*86400 as integer)
```

For DB2:

```
timestampdiff( 2, char((current timestamp - current timezone)</tt> - <tt>placed_in_queue_at))
```

_time_in_same_queue

Same as `_time_in_queue` except that it is based on the `MovedToQueueAt` property, which is updated only when the interaction is placed into a queue that it has not been in before.

Provides a unified method of calculating the amount of time that the interaction has spent in the current queue. It calculates the difference in seconds between the current time in UTC and the attribute `MovedToQueueAt`.

Usage

```
_time_in_same_queue()
```

Translations Performed by Database

For Oracle 9 and 10:

```
((cast(SYS_EXTRACT_UTC(SYSTIMESTAMP) as date) - moved_to_queue_at)*86400)
```

For Microsoft SQL:

```
cast(cast((getutcdate() - moved_to_queue_at) as float)*86400 as integer)
```

For DB2:

```
timestampdiff( 2, char((current timestamp - current timezone) - moved_to_queue_at))
```

_empty

Usage

```
_empty(string expression)
```

The result is a logical expression that is true if the value of the string expression is either null or an empty string, and is false otherwise.

Translations Performed by Database

For Oracle:

```
((value) is null)
```

For all other database platforms:

```
(((value) is null) or ((value)=''))
```

The reason the Oracle condition is different is that Oracle interprets an empty string as null, and a condition like `(anything=null)` is always false.

_not_empty

Usage

```
_not_empty(string expression)
```

The result is a logical expression that is false if the value of the string expression is either null or an empty string, and is true otherwise.

Translations Performed by Database

For Oracle:

```
((value) is not null)
```

For all other database platforms:

```
(((value) is not null) and (not ((value)='')))
```