



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

eServices Interaction Properties Reference Manual

Genesys Engage Digital (eServices) 8.1.4

12/29/2021

Table of Contents

| | |
|--|----------|
| eServices Interaction Properties Reference Manual | 3 |
| System Properties | 5 |
| Business Properties | 9 |
| Custom Properties | 11 |
| Business Attributes | 13 |
| Translations | 16 |

eServices Interaction Properties Reference Manual

Genesys eServices interactions have a number of properties that take the form of key-value pairs. This document lists these properties and provides some information about them, including whether it is safe to change them.

Since you can set up your eServices system to change some of these properties as the interaction moves through the system, it is particularly important to know that some properties are safe to change and others are not. For example, you can change interaction properties by using the `Update`, `UpdateBusinessData`, and `UpdateInteractionData` functions in a routing strategy. But you should be very careful when considering using these functions to change any interaction property.

There are three types of properties that can be defined in terms of the following two characteristics

- Whether they exist as independent fields in the interactions table. If they do, you can refer to them when defining conditions, orders, and segments in Views in Business Processes.
- Whether they are used by Genesys media servers.

The following table summarizes this classification and specifies whether it is safe for users to change the properties.

Important

When this document says that you must not change an interaction property, that means that you must not change the value of existing properties *and* that you must not create a new object with this name.

Three Types of Interaction Property

| Type | Independent Field | Used by Genesys Media Server | Change OK |
|----------|----------------------|------------------------------|------------------------------------|
| System | Yes | No | No, except ExternalID and ParentID |
| Business | Mostly no; see below | Yes | OK if not used by media server |
| Custom | No | No | Yes |

Here is further information on these three property types:

- **System properties.** These are maintained solely by Interaction Server, as independent fields in the interactions table. The user cannot change them, with the two exceptions listed in the preceding table. The definition of conditions, orders, and segments in Views can refer to them. Examples: InteractionID, MediaType.
- **Business properties.** The values of these are set by media servers (including both Genesys and custom media servers), and these properties may be used by the media servers at later points in processing. Therefore you must not change the ones that are used by any media server in your solution. Some are **stored as independent fields** in the interactions table but most are stored in the FlexibleProperties field. The definition of conditions, orders, and segments in Views can refer to ServiceObjective and Priority, which are the only ones that are stored as independent fields. They can also refer to the other Business properties if you **create custom fields** that correspond to them.

Business properties contrast with **business attributes**, which are objects in the configuration database.

- **Custom properties.** These exist only if you define them. Media servers do not use them. The definition of conditions and orders in Views can refer to them.

Important

In setting conditions for Views and snapshots, besides directly using some interaction properties types, you can also use a set of functions called **Translations**. These provide database manipulation tools that are independent of the underlying database.

Interaction properties show up in several places:

- Interaction Server logs show all of them.
- Some IRD objects (for example, UpdateBusinessData) use them as parameters and display them in drop-down lists.

System Properties

System properties must not be changed, with the two exceptions listed in the [Three Types of Interaction Property](#) table. You can use these properties on the **Condition**, **Order**, and **Segmentation** tabs of Views in Business Processes, except that properties with Timestamp data type cannot be used on the Segmentation tab.

This table lists the following about system properties:

- Property name as it appears in protocol messages
- Field name in the interactions table, if the property is stored as an independent field
- Data type
- Short description

| Name | Name in Interactions Table | Type | Description |
|--------------------|----------------------------|-----------|---|
| AbandonedAt | abandoned_at | Timestamp | Date and time that the media server set the <code>IsOnline</code> attribute to 0. Simplifies calculation of some statistics. If the interaction is still online, this attribute is not set. |
| DeliveredAt | delivered_at | Timestamp | Date and time that the interaction was first offered to the agent |
| ExternalId | external_id | String | External interaction identifier (examples: chat session ID, e-mail mime ID) |
| FlexibleProperties | flexible_properties | Binary | Stores attached data and most business properties (see Business Properties). |
| InQueues | destinations | String | Suggested destinations for the interaction (if provided by URS) |
| InteractionId | id | String | Record identifier |
| InteractionState | state | Integer | 0 = queued 1 = cached 2 = being processed by URS 3 = being handled by agent |

| Name | Name in Interactions Table | Type | Description |
|--------------------|----------------------------|-----------|--|
| InteractionSubtype | subtype | String | Defined as Business Attribute in Configuration Server |
| InteractionType | type | String | Defined as Business Attribute in Configuration Server |
| IsLocked | is_locked | Integer | 0 = unlocked 1 = locked |
| IsOnline | is_online | Integer | 0 = offline 1 = online This property applies to any media type (for e-mail, the value is always 0). |
| MediaType | media_type | String | Defined as Business Attribute in Configuration Server |
| MovedToQueueAt | moved_to_queue_at | Timestamp | Date and time that the interaction was first moved to a queue. If the interaction leaves the queue, then returns to it, the value of this property remains the same; that is, it shows the earliest time that the interaction entered the queue, without regard for later entry to or exit from the queue. |
| OutQueues | destinations | String | Suggested destinations for reply |
| ParentId | parent_id | String | Identifier of the parent interaction in the UCS database |
| PlacedInQueueAt | placed_in_queue_at | Timestamp | Date and time that the interaction was placed in the current queue, regardless of whether the interaction was in this queue previously |
| PlaceInQueueSeq | place_in_queue_seq | Integer | Event sequence number, denoting the chronological order of the last EventPlacedInQueue (for the interaction) |

| Name | Name in Interactions Table | Type | Description |
|---------------------|----------------------------|-----------|--|
| Queue | queue | String | Name of the queue that the interaction is in |
| ReceivedAt | received_at | Timestamp | Date and time that the media server received the interaction. If not provided by media server, date and time of submission to Interaction Server (same as SubmittedAt). |
| ScheduledAt | scheduled_at | Timestamp | Date and time before which the interaction must not be processed. See “Setting the ScheduledAt Property” in the “Creating Business Process Objects” chapter of the Universal Routing 8.1 Business Process User's Guide . |
| SubmittedAt | submitted_at | Timestamp | Date and time that the interaction was submitted to Interaction Server |
| SubmittedBy | submitted_by | String | Name of the client application that submitted the interaction |
| SubmitSeq | submit_seq | Integer | Event sequence number, denoting the chronological order of EventInteractionSubmitted (for the interaction) |
| TenantId | tenant_id | Integer | Tenant associated with the interaction |
| Workbin | workbin | String | Indicates that interaction is in a workbin |
| WorkbinAgentGroupId | agent_group_id | String | One of four properties specifying the ID and type of the workbin that the interaction is in. Only one of the four properties is present. |
| WorkbinAgentId | agent_id | String | See WorkbinAgentGroupId. |
| WorkbinPlaceGroupId | place_group_id | String | See WorkbinAgentGroupId. |

| Name | Name in Interactions Table | Type | Description |
|----------------|----------------------------|-----------|--|
| WorkbinPlaceId | place_id | String | See WorkbinAgentGroupId. |
| AbandonedAt | abandoned_at | Timestamp | Date and time that the media server set the IsOnline attribute to 0. Simplifies calculation of some statistics. If the interaction is still online, this attribute is not set. |
| HeldAt | held_at | Timestamp | Time and date, set by Interaction Server, that the interaction was put on hold. |
| CompletedAt | completed_at | Timestamp | Date and time, set by Interaction Server, that the interaction was first placed into one of the queue specified in the Interaction Server completed-queues option. |
| AssignedAt | assigned_at | Timestamp | Date and time, set by Interaction Server, that the interaction was last delivered to the resource. |
| AssignedTo | assigned_to | String | Employee ID of the agent to whom the interaction was last delivered. If the agent is anonymous, concat('@',place name) is used instead. |

Business Properties

Business Properties are set by media servers, so it is safe to change these properties only if they are not used by any media server in your solution.

Business Properties are stored in two ways. The two properties listed in the following table are stored as independent fields, so you can use them on the Condition and Order tabs of Views in Business Processes.

This table lists the following about system properties:

- Property name as it appears in protocol messages
- Field name in the interactions table, if the property is stored as an independent field
- Data type
- Short description

Business Properties Stored as Independent Fields

| Name | Name In Interactions Table | Type | Description |
|------------------|----------------------------|---------|---|
| ServiceObjective | service_objective | Integer | Time objective for servicing the interaction. The contact center may define a service objective for each combination of customer segment, service type, and media type. |
| Priority | priority | Integer | Indicates whether e-mail should receive special processing |

All other Business Properties, listed in the following table, are stored in the `flexible_properties` field. To use them on the Condition, Order, or Segmentation tabs of Views in Business Processes, you must **create Custom Properties** that correspond to them.

Business Properties Stored in FlexibleProperties

| Name | Type | Description |
|--------|--------|---|
| Caseld | String | Case identifier. Use and meaning to be defined by user. |

| Name | Type | Description |
|-----------------|--------|--|
| CategoryId | String | Category identifier obtained by routing strategy request for e-mail classification |
| ContactId | String | Customer identifier in the UCS database. Provided by UCS. |
| DispositionCode | String | Code for moving the interaction somewhere else |
| FromAddress | String | Taken from e-mail interaction |
| FromPersonal | String | Name of person who sent the interaction |
| Header_* | String | Content of the header of an e-mail. Do not change. |
| Mailbox | String | Mailbox of addressee |
| ReasonCode | String | Code for reason for the operation that caused the event; for example, normal, autoresponse, sent, forwarded, or redirected. Set by routing strategy Stop object. |
| CustomerSegment | String | Code for the customer's revenue potential; for example, Gold, Silver, Bronze. May be assigned as a result of database lookup based on sender name. Routing strategy sets this property. |
| ServiceType | String | Code for type of service being requested; for example, Sales, Service, Information. Routing strategy sets this property. |
| Subject | String | Taken from e-mail interaction |
| To | String | Destination e-mail address of an e-mail or web form. Do not change. This property is required if you use the Chat Transcript object in routing strategies. It supplies the value for the To field of the outbound e-mail that sends the chat transcript. One way to give this property a value is in a routing strategy; for details, see "Chat Transcript" under "eServices (Multimedia) Objects" in the "Interaction Routing Designer Objects" chapter of the Universal Routing 8.1 Reference Manual . |

Custom Properties

You can create new interaction properties (fields). The data type of these custom properties can be timestamp, string, or number. You can use these properties on the Condition, Order, and Segmentation tabs of Views in Business Processes, except that properties with Timestamp data type cannot be used on the Segmentation tab.

Important

In a multitenant environment, the configuration of interaction custom properties in one tenant applies to all tenants. This means that the values of custom properties are saved in separate corresponding database fields for all interactions, regardless of whether the interactions belong to the tenant in which the custom properties are configured.

Configuring a custom interaction property

1. Decide on an attached data key that will be the source of the content of the custom property.
2. Create a new field directly in the interactions database.

Important

Data type varchar(max) is not supported for custom fields.

3. Create a new Business Attribute:
 - Name = InteractionCustomProperties
 - Display name = Interaction Custom Properties
 - Type = Custom

If such an attribute already exists go to the next step.

4. Expand Interaction Custom Properties and open its Attribute values.
5. Give it an Attribute Value, with a name exactly matching the attached data key name that you decided on in Step 1. The matching is case sensitive (you can create a separate display name).
6. In your new attribute value, go to the Annex tab and create a section called translation.
7. In the new translation section, create an option called translate-to, with its value duplicating the name of the new field you created in Step 2.

Next Steps

You can now use the new custom property to attach data to an interaction and to define conditions

and orders of Views and snapshots.

Notes

You should be aware of the following points when defining custom interaction properties:

- While Interaction Server allows defining custom property names that contain spaces in their name, and will correctly map these properties to the custom database fields, it does not convert these property names into custom field names when they are used in the definition of a view condition, view order, snapshot condition or snapshot order.
- Genesys DB Server does not support custom fields of type varchar(max).
- If you specify a custom field as not null, you must ensure that applications always provide some data to that field upon creation of an interaction (RequestSubmit). If no data is provided, the request will fail because Interaction Server sends NULL for empty fields, and that will be rejected by the DBMS. This also means that the default value trigger for such fields cannot be used.

As a workaround, you can create fields in the interactions database without mapping them to custom properties of Interaction Server. Such fields are hidden from Interaction Server but can be used by third-party applications.

Business Attributes

Business Attributes differ from interaction properties, although the two share some names (for example, MediaType and Case ID).

- Interaction properties are fields in the Interactions table. Their main purpose is to identify and describe interactions during processing.
- Business Attributes are objects in the configuration database. They have two purposes.
 - To represent data concerning areas other than traditional telephony. Each Business Attribute contains a range of possible values for a data category. In this sense, Business Attributes are like a dictionary.
 - To convey data between software components that have no other means of communication. Examples are Category Structure and Screening Rules. These are among the Business Attributes that you must not change in Configuration Manager.

Category Structure is an example of a Business Attribute that users must not change. You create and edit categories in Knowledge Manager. The categories that you create also appear in Configuration Manager, in Resources > Business Attributes > Category Structure. It is possible to select a category in Configuration Manager, open its Properties dialog box, and make changes as you would to any configuration object. But this could create severe problems and the system would probably not operate properly when trying to handle this category. Knowledge Manager is the only safe tool to use for creating and editing categories.

The following table lists the Business Attributes and Business Attribute Values that are relevant to eServices but are not represented in the interactions table.

Business Attributes Not in the interactions Table

| Name | Change |
|---------------------------------|------------|
| CategoryStructure | Prohibited |
| ContactAttributes | Prohibited |
| ContactAttributes/AccountNumber | Allowed |
| ContactAttributes/LastName | Prohibited |
| ContactAttributes/PhoneNumber | Prohibited |
| ContactAttributes/PIN | Allowed |

| Name | Change |
|-------------------------|------------|
| ContactAttributes/Title | Prohibited |
| EmailAccounts | Allowed |
| InteractionAttributes | Prohibited |
| Language | Allowed |
| RootInteractionID | Prohibited |
| ScreeningRules | Prohibited |
| StopProcessingReason | Allowed |

The next table repeats three of the Business Attributes from the preceding table for which change is allowed, and adds information. Note that the first two of these Business Attributes must be present for eServices to work properly.

Business Attributes, Change Allowed

| Name | Default Value | Comment |
|---------------|---------------|--|
| EmailAccounts | None | <p>The eServices configuration wizard creates two values: one for the account that E-mail Server pulls e-mails from in order to bring them into the eServices system, and one for an external agent.</p> <p>You may want to use Configuration Manager to create additional values, for example if you want additional e-mail addresses to be available for Forward and Redirect objects in routing strategies.</p> <p>Configure the address for this value on the Annex tab > General section > address option. The address must comply with RFC2822 and therefore it must be encoded according to RFC2047.</p> <p>Examples: legal@mycompany.com ""Legal Dpt"" <legal@mycompany.com> Type = custom</p> |
| Language | English | Used by Knowledge Manager (along with |

| Name | Default Value | Comment |
|----------------------|---|--|
| | | Tenant) to group category trees. You must add the value unknown if you want to train a model that classifies by language. See See Training and See Notes on Language. |
| StopProcessingReason | AutoResponded Forwarded Normal Re-directed Sent Terminated | Must be selected in the IRD Stop Interaction object. You can manually add other values by creating them as Attribute Values of StopProcessingReason, under Business Attributes in Configuration Manager. |

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>(date2)) * 86400)
```

For Microsoft SQL:

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

For DB2:

```
timestampdiff( 2, char((date1) - <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>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)=''))
```