GENESYS™

Developer's Guide

Context Services 8.5.2

12/30/2021

Table of Contents

Context Services Developer's Guide

These developer pages, primarily intended for programmers developing strategies for contact center agents, assume that you have a basic understanding of:

- Computer-telephony integration (CTI): concepts, processes, terminology, and applications.
- Network design and operation.
- Your own network configurations.

Introduction

This developer's guide covers the writing and the optimization of your applications on top of the Context Services. Representations, requests, and responses are detailed in the API Reference page. You should use this developer guide to learn about the operations and representations used in this REST API. Developer pages are intended to help you to:

- Understand the design of the Context Services
- Get details to optimize your application's architecture
- Give general directions to your implementation on top of this product.

Important
If pages are missing information or not helpful enough, use the comment form at the bottom of the page to submit questions and feedbacks.

Using the Context Services

Your application should use Context Services to manage conversation data (or services) with their nested states and tasks. You can use these services to track customer conversations across channels and manage smart transitions between those.

For instance, let's consider a simple use case. If your customer interacts through an IVR with your call center, you can record information in one or more services. In case your customer is disconnected and calls again, or even makes an attempt through another channel, you can recover the conversation and the customer won't have to repeat information or to go through the same steps twice.

You can also use the Context Services to learn from your customers' conversation. Let's imagine, for instance that only 75% of customers do not complete a conversation: Context Services lets you know where they decide to quit and also provide statistics through the Pulse dashboard.

Context Services makes the storage of those conversations simple and easy. Your application can manage service data through JSON queries, which start and stop the services and their inner states and tasks.

Important

Note the following terminology:

- *GMS/CS* is used when describing the Context Services capabilities in relation with some components (DataDepot or Cassandra database) of the GMS platform.

- *UCS/CS* is used when describing former Context Services capabilities in relation with some components of the Universal Contact Server, only available for backward compatibility.

See the Architecture page for further details.

New in This Document

The following topics have been added or changed in the CS 8.5.205 release:

- The section Bulk Updates of Service, State, Task was added.

Implementation Changes in 8.5

In this page, you will learn about the major differences between the 8.1 and 8.5 versions of the Context Services API. In addition to architecture changes, some features were deprecated and other were improved to increase the API usability and performance results.

A New Architecture

In 8.5, Context Services is available on top of Genesys Mobile Services (GMS). This means that your application no longer needs a connection to the Universal Contact Server (UCS) to manage services, states, and tasks. Data related to customer management (contact information, profiles, interactions) are still available for backward compatibility in UCS. See Architecture for details.

So, if you are upgrading from a version anterior to 8.5, you must read the migration instructions.

Multi-Tenancy

You will find instructions in the User's Guide to configure your Context Services application in a multi-tenant environment. Then, you just add two headers to handle the Tenant information in your queries (click here for details).

If you do not need multi-tenancy, you don't have anything to change in your application.

API URL

Context Services is now available on top of GMS. You should access the context services through the /genesys/1/cs/ base URL, as for example:

`POST http://localhost:8080/genesys/1/cs/services/start`

You can still access Customer Profiles in UCS. To simplify your deployment, you should deploy your UCS application with the `cview/base_url` option set to /genesys/1/cs as detailed in the migration page.

Customer Profile API

Customer Profiles includes Contact Profiles and Interactions APIs. This UCS API is not available in the new Context Services API developed on top of GMS. It remains in UCS, either exposed as HTTP or ESP protocol, as detailed in the new architecture page.

This feature is optional. You no longer need the Customer Profile API and a connection to UCS to use Context Services. See the migration page for further details.

This API is not concerned with major changes such as universal unique identifiers or new JSON extensions. The management of this API did not change in 8.5 and does not differ from the former latest 8.1 version, except for a few deprecated queries (see the list here). For a detailed list of resources and queries, refer to the Context Services API Reference.

Resource Identifiers

Services, States, Tasks resources use Universal Unique Identifiers(UUIDs), instead of former legacy numbers (64bit or 32bit).

Important

This change does not concern the Customer Profile API.

Extensions Metadata and Schema Management

Extensions are now JSON key-value pairs in the Services API. You no longer have to manage extension schemas for objects of type service, state, and task. The JSON value of an extension ensures the backward compatibility for reading and writing service resources.

- You can no longer remove a set of service, states, or task extension data associated with a specific schema in a single request.
- Your application is responsible for managing any strong typing approach, for instance, to define which data is mandatory or not.
- Your application is responsible for maintaining extension uniqueness across the database.
- Your application can no longer use queries which include the "by unique" criteria.

You can get a list of deprecated queries here.

Exporting Capabilities

Context Services now includes exporting capabilities, available through new Services API operations. You can either export a list of service IDs (retrieved as a JSON object), or retrieve service data in a .CVS file. Additionally, the new operations include filters to refine your export action.

For further details, refer to the Export features page.

Purging Capabilities

In 8.5, Context Services provides purging capabilities based on the Time to Live (TTL) criteria and assigns an expiration date to any data added to the Cassandra database. You can perform purging tasks through the Purge Service query or schedule tasks in the Configuration Server, as detailed in the user guide.

Role-Based Access Control

The role-based access control is simplified. You need Administrator or Supervisor privileges to handle services, states, and tasks. Click here to get more information about these privileges.

Important

Role-based access control did not change between 8.1 and 8.5 for any customer profile management.

Business Attributes Mapping

You will now use enumerated names, available from the CME/GAX enumerator instead of numbers. Context Services automatically validates enumerated names, then converts these names into DBIDs for further storage.

Additionally, the Context Services can transform `from/to` and `Long/String` properties during the serialization and de-serialization process.

Important

In a single-tenant environment, this configuration is also compatible with UCS/CS.

Event Submissions

Your application now deals with `Start, End, Transition,` and `Associate` for Service, State, Task events. These events include the following properties, in addition to optional attached key-value pairs:

- `contact_key`
- `customer_id`

- `service_id`

- `state_id`

- `task_id`

- `service_type` or `state_type` or `task_type`

- `previous_state_id`

- `est_duration`

- `disposition`

- `disposition_desc`

- `status` (which indicates if the complete action occurred)

> ### Important
> The other fields of the `start/completed` events are stored as part of `ActivityEvent` events.

Deprecated Queries

The following queries are no longer available, due to architecture or data changes (read the previous sections for further details.)

```
PUT /services/${service id}/extensions/${ext name}/by/unique
PUT /services/${service id}/extensions/${ext name}/delete/by/unique
PUT /services/${service id}/state/${state id}extensions/${ext name}/by/unique
PUT /services/${service id}/states/${state id}/extensions/${ext name}/delete/by/unique
PUT /services/${service id}/task/${task id}extensions/${ext name}/by/unique
PUT /services/${service id}/task/${task id}/extensions/${ext name}/delete/by/unique
POST /metadata/services/extensions
GET /metadata/services/extensions
POST /metadata/states/extensions
GET /metadata/states/extensions
POST /metadata/tasks/extensions
GET /metadata/tasks/extensions
DELETE /metadata/services/extensions/${extension-name}}
DELETE /metadata/states/extensions/${extension-name}
DELETE /metadata/tasks/extensions/${extension-name}
```

New Interfaces

Thanks to the new architecture, we implemented two additional interfaces for you in the GMS Service Management Interface:

- With the Customer Journey GUI, you can learn about a customer and his or her navigation through the services.
- With the Context Service Interface GUI, you can create and edit services, states, and tasks.

These tools are intended for developers and administrators. (**Tell me why.**

These interfaces are built to search for profiles, services, states, and tasks based on ID information or UCS information. They do not include all the search abilities that are available in typical agent interfaces.
)

Architecture

In this page, you will learn about the new 8.5 architecture of Context Services, now built on top of Genesys Mobile Services (GMS) for the service part, and on top of Universal Contact Server (UCS) for the customer data part.

New Architecture

Before 8.5, Universal Contact Server (UCS) managed and stored all the Context Services' resources and requests. In 8.5, the data is split between Genesys Mobile Services and Universal Contact Server to fulfill Genesys models.

If your application is using customers and interactions, you can still get them in UCS, but service data is now stored in the Cassandra databases of the GMS Cluster.
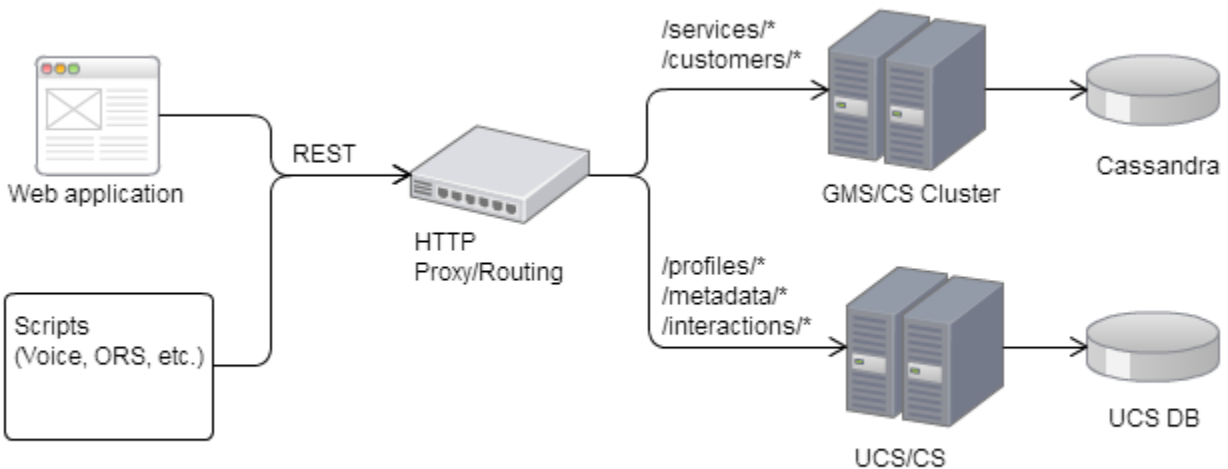
New deployment of Context Services, based on a GMS Cluster.

Thanks to this architecture, you can use the Context Services independently from Universal Contact Server if you don't need to handle profiles in UCS. This architecture also ensures the backward compatibility if your application still need to manage these profiles.

- For the GMS/CS part, you benefit from the GMS Cluster's scalability, other GMS features, and new integration tools, such as Pulse integration and Customer Journey.

- For the UCS part, pages related to profile management inform you to enable profiles in notes.

Deploying your Application

- You must configure your proxy to redirect correctly the applications' REST queries.

- If you are moving from 8.1 to 8.5, you must migrate your services.

Click here to learn how.

GMS/CS also supports multi-tenancy. You will find instructions in the User's Guide to configure your GMS/CS application in a multi-tenant environment. Then, you just add two headers to handle the Tenant information in your queries (click here for details).

Services, States, and Tasks

This page gives guidelines for managing Service information with Context Services. In 8.5, all the service management is handled on top of GMS and is simplified to facilitate the whole data process.

About the Service, State, and Task Resources

Services are customer commitments defined by the business application (IVR, Orchestration, Agent Desktop, etc.) which interacts with the customer. Each service potentially spans multiple interactions over a variety of media channels and should link to a Customer Profile as soon as it is created or retrieved through identification operations (read Profiles and Identification.)

The Context Services REST API uses Service, State, and Task resources to manage and store the context information of your application. Basically, the Service resource is equivalent a top-level container associated with an overall commitment, which can be divided into a set of States to transition from one to another. These additional states can be divided into tasks.
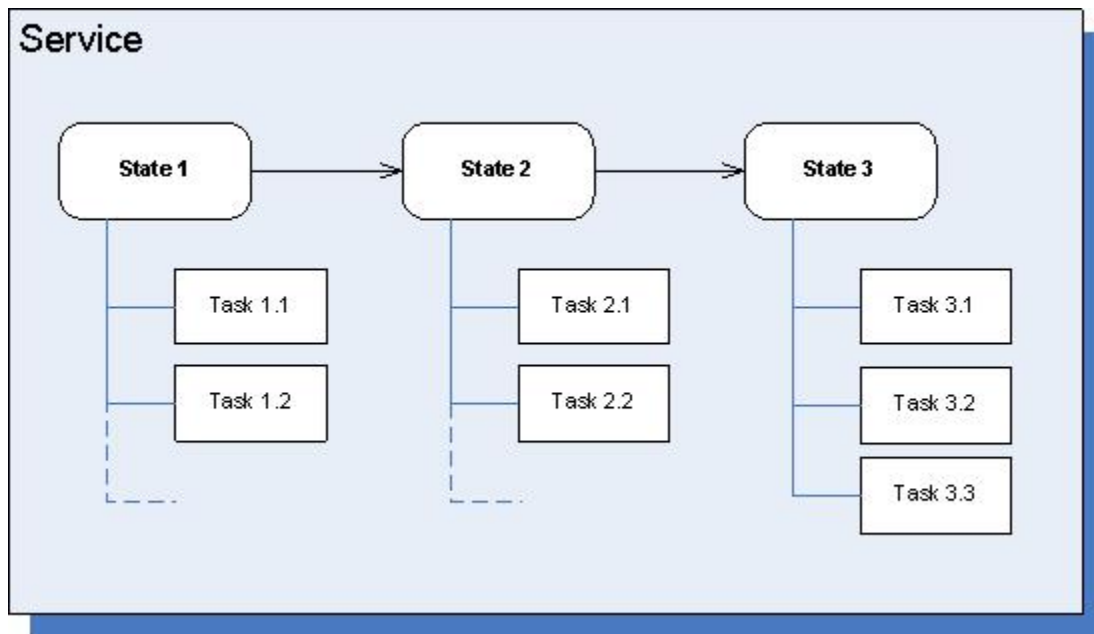
Consider, for instance, an application that is a web-based interface, and that includes several online services, such as 'Booking a hotel reservation'. The service 'Booking a hotel reservation' is in charge of collecting information for the reservation.

- State 1: Collect Hotel Search Information
  - Task 1: Collect Time Information (arrival, departure)
  - Task 2: Collect Localisation Information

- Task 3: Collect Hotel Criteria

- State 2: Get Proposals

  - Task 1: Search offers in the database

  - Task 2: Propose offers

  - Task 3: Get Detailed Information about the offer

- State 3: Validate Proposal

  - Task 1: Get customer approval

  - Task 2: Make payment

  - Task 3: Validate reservation in the system

  - Task 4: Send bills and additional details by email

  - Task 5: Collect customer feedback

- And so on.

If a customer starts interacting with the service, the application creates a new service resource to manage the service's context data, and then nested state and task resources to manage further states and tasks' context data.

Services, States, and Tasks contents

The standard content of these resources is formal core information, as described in the related Service, State, Task pages, to determine:

- When the given service, state, or task started;

- Whether it is active or completed;

- Which interactions or customer are related to the given instance.

For each type of resources, the Context Services provide you with a set of operations which manage this basic data. For instance, in the case of a service, standard use cases imply that your application should:

1. Start the service,

2. Associate the service with a customer ID—see Anonymous Service for further details;

3. Start and complete similarly states and tasks—see List of Operations;

4. Complete the service once all the nested states and tasks are completed.

Important

Your application is fully responsible for managing the status changes for the nested states and tasks. In other words, if you want to complete a given service, you must also complete the nested states and tasks.

Active Resources

A service, state, or task is active if a customer is still interacting with it. In that case, the service, state, or task is started, but not complete. Once the resource is completed, it is no longer part of the active list, but part of the completed list.

> ### Important
> Read also Query Services/States/Tasks

.

Extensions

You can use extensions to record additional data related to the management of services, states, or tasks. In our introduction sample of 'Booking a hotel reservation', this would represent all the information collected through the service. If at any time the customer is disconnected and reconnects, you can fetch this information in Context Services and recover the conversation.

To add or update extension data, you simply add key-value pairs to REST queries where the key is a string, and the value is some JSON data (for example, a string, a JSON array, or a JSON object).

> ### Tip
> Context Services ensures backward compatibility with 8.1. If you migrate from 8.1 to 8.5, you do not need to modify your service queries, and you no longer need to handle schemas for services and their nested resources. You will find related instructions here.

In the API reference, you can see whether you can add or update extensions if the following attribute is available:

Name	Type	Mandatory	Description
<extension key>	Any JSON type	No	Service attached data as key-value pairs. You can add as many key-value pairs as needed.

Basic Service, State, and Tasks Management

Operations and resources in this section are part of Service Operations and its subcategories.

Start the Service

1. Start Service: first step in your service management. You create a service instance each time that a new information context needs to be created. (In our example, each time a customer enters in the Booking reservation service, UCS creates the core service information, including a service ID returned as a result of this operation.)

   - If you have no information to create or identify the customer, your service is anonymous. In that case, use a contact key.

   - When you start the service, you pass in the operation's body the Service Start Event which describes the start information.

2. Associate Service: To use later, once you have a customer ID to associate with your service. To get a customer ID, you need to retrieve profile information (see List of Profile Operations).

The following operation starts a new service with a contact key:

```
POST /services/start
{
  "timestamp": "2009-05-12T12:05:12.145Z",
  "interaction_id": "123ABCAADFJ1259ACF",
  "application_type":400,
  "application_id":40,
  "est_duration":60,
  "contact_key":"42",
  "service_type":100,
  "media_type": 1,
  "resource_id": 5005,
  "resource_type": 2,
  "disposition": 10,
  "coupon": {
          "coupon_name": "DISCOUNTCODE15"
  },
  "satisfaction": {
    "score": 85,
    "agentID": 2025
  },
  "relatedOffers": [
          {
           "offer_name":"VIP credit card black ed.",
           "type":9,
           "comments":"proposed to all client"
          },
          {
           "offer_name":"3 times payment GOLD",
           "type":4,
           "comments":"limited offer"
          }
          ,{
           "offer_name":"life insurance",
           "type":3,
           "comments":"healt check to be done before approval"
          }
  ]
}
```

In the above query, `coupon`, `satisfaction`, and `relatedOffers` attributes are extensions.

Manage States or Tasks for a given Service

You can use both States and Tasks, or Tasks only.

1. Start State or Start Task: In the corresponding Start Event, you must specify to which service the state or task belongs by filling the `service_id` parameter.

2. Perform State Transition: if your service contains several states, you can perform state transition instead of completing a state and starting a new state.

3. The state transition does not complete the tasks which belong to the completed state. Your application must complete them before performing this operation.

The following example shows a state transition:

```
POST /services/735692/states/transition
{
  "timestamp": "2009-05-07T12:05:20.157",
  "session_id": "11000ABC-80236C1A-1010",
  "interaction_id": "123ABC908ABFFD8080",
  "from": {
    "state_id": 1001,
    "disposition": 1,
    "disposition_desc": "SUCCESS",
    "Feedback":
      { "FeedbackType":"survey","rating":7,
        "notes":"warm welcome at frontdesk, thanks for the nice trip"
      },
    "Satisfaction": [
    {
      "rating":2,
      "pertinence":8,
      "usefull":true,
      "place":"Terranova mexico resort"
    },
    {
      "rating":8,
      "pertinence":4,
      "usefull":false,
      "place":"Fancy resort Paris"
    }
    ]
  },
  "to": {
    "state_type": 8,
    "est_duration": 500,
    "Sponsoring": { "Rank":"first","expire":7,
      "notes":"give customer free meal" }
  }
}
```

Query the Services/States/Tasks for a Given Profile

In Query Services, Query States, and Query Tasks operations, you can query lists of active or completed resources, by filtering in the URL the `active` or `completed` status of the resources. For instance, in Query Services:

- Active Services: GET `/customers/${customer_id}/services/active`

- Completed Services: GET `/customers/${customer_id}/services/completed`

In addition, the Query Services and Query States operations enable to retrieve the nested states and/or tasks, within the results. For instance, the following query operation returns the active services within their active states associated with the customer profile *ABC1234*.
Operation

```
GET /customers/ABC1234/services/active?active_states=true
```

Response

```
[ // returned in an array
 { "customer_id": "ABC1234",
  "service_id": 4692834,
  "est_duration": 86400,
  "started": {
   "timestamp": "2009-05-07T12:05:20.157",
   // additional Start Event fields
 },
 "active_states":
  [// included given specification of "results" attribute
   { // array of one or more State objects
    "state_id": 5005,
    "state_type": 8, // service delivery
    "started": {
      "timestamp": "2009-05-07T12:08:53.298",
      // additional Start Event fields
    }
   }
  ]
}]
```

Complete Service/State/Task

When your customer stops interacting with the given service, state, or task, you must complete this resource and mark it as terminated in the UCS database. This enables you to filter the result of query operations based on the resource status (as described in the Active Resources section).

- You are responsible for performing the Complete Service, Complete State, Complete Task operations for any service, state, or task that you started.

- These operations apply only to the resource specified in the operation's parameter and they do not modify the status of the nested states and tasks, if any.

The only case which does not force you to explicitly complete a state with the Complete State operation, is Perform State Transition, which completes the given state then starts a new state.

> **Tip**
>
> To make sure that you correctly completed the states and tasks of a given service, use the Query States and Query Tasks operation with active filters to check that no resources remain active.

Auto-Complete Service

Introduced in 8.5.111

Instead of completing a service manually, you can enable auto-completion by setting the auto-complete-enabled option to `true` and by configuring an auto-complete-after time in your GMS configuration. In that scenario, the system will wait for the given `auto-complete-after` time to complete the service after its last update and/or the last update to its nested states and tasks.

If the feature is enabled, you can also choose to override this auto-complete-after time by passing this value in the `auto-complete-after` parameter of your Start Event at the service start.

Additionally, if the auto-complete feature is enabled and if you neither configure a default value nor pass one at service creation, you will have to complete the service with a REST query to make it end.

Improve Performance while Querying Data Associated with a Customer

Introduced in 8.5.111

When you query customer or services resources, you can speed up the results by limiting the number of services to be returned. To implement this limit, you can add the parameter `max_services` to your GET query and only the last created services will be returned.

This parameter can be added to the following queries, detailed in the Query Services page:

- GET /genesys/1/cs/customers/${customer_id}/services
- GET /genesys/1/cs/customers/${customer_id}/services/active
- GET /genesys/1/cs/customers/${customer_id}/services/completed
- GET /genesys/1/cs/services/anonymous/${contact_key}
- GET /genesys/1/cs/services/anonymous/${contact_key}/active
- GET /genesys/1/cs/services/anonymous/${contact_key}/completed

Bulk Updates of Service, State, Task

Introduced in 8.5.200

To perform a bulk update, use the composite start query and specify `update_extensions=true` in the start event. Instead of creating a new object, the operation will update the provided extensions of the given objects. This means that only the specified extensions will be updated. For example, if your service's extensions include the `Satisfaction` extension, and if you do not provide this extension in the query, the extension will neither be updated nor removed from the service.

For details and examples, refer to Composite Start and Bulk Update.

Anonymous Service

This page details the management operations for services with no identified contacts.

Definition

An **anonymous** service is a service which is assigned to an anonymous customer. The customer is still unknown, so no customer ID is assigned to the service. Your application is in charge of assigning this customer ID as soon as you identify the customer. Read also Services, States, and Tasks.

Use Case

In many situations, you can identify the customer prior to the creation of the service, which ensures the possibility of adding the customer ID to the service in the Start Service operation. For example, if the customer explicitly logs in your website before your web application invokes the service, or if your IVR identifies the customer and then chooses a service. In those cases, your application can specify the customer ID at the service creation.

However, in other cases, your application may start the service before the customer is identified. Therefore, if your application cannot specify the customer's ID at the service creation, the service is **anonymous**. Let's consider a customer who is filling out an order on a web site before he or she has explicitly logged in, or a preliminary service delivered in the IVR before the customer is prompted for identity information. In these cases, the application is not able to provide the customer identifier.

The Contact Key

You can create an anonymous service with the Start Service operation. In that case, even if the customer is not identified, your application must pass a contact key, based on the current information available. The "contact key" is supplied at the service creation. Then, you can query the service even if it is anonymous with Query Anonymous Services.

For example, the following information can be used as contact keys: e-mail address, phone number, lastname+firstname.

Related Operations

- Associate Service
- Start Service

- Query Anonymous Services

Business Attributes

This page introduces Business Attributes in Context Services, lists the types of useful business attributes, lists all the method used to manage business attributes.

Definition

Management Framework creates and manages enumerations known as Business Attributes. These attributes are modeled in Context Services as integers which represent the Management Framework DB ID for a given enumerated value. For example, an organization might define the "service type" Business Attribute, made of two enumerated values:

- "New Account"(DB ID = 1);
- "Bill Payment" (DB ID = 2).

Your application is responsible for the further use of those values and should only use the appropriate business attribute values.

Business Attributes in Context Services

The following specific fields are validated against specified mapped Business Attributes in the Configuration Server:

- service_type
- state_type
- task_type
- application_type
- resource_type
- media_type
- disposition

Tell me how I can configure these attributes.
You can find configuration information in two sections of the User's Guide:

- How to map Context Services with Business Attributes.
- The options' reference for GMS/CS Business Attributes.

)

Their use concerns the Service, State, and Task representations. Context Services automatically rejects wrong unknown enumerated values, and returns a proper response which directs you to use the valid enumerated values of the configured Business Attributes. The system includes a service for returning information on attributes mapped to the Configuration Server attributes, including information on the DB ID, unique name, display name, and description for all values of the mapped Business Attributes.

> ## Important
>
> - Context Services only validates incoming data against the current Business Attribute definitions. It does not guarantee referential integrity over time. More specifically, both Genesys Administrator and Configuration Manager allow the modification of the Business Attributes definitions over time.
>
> - When Business Attributes are deleted, this operation does not modify the historical service records which may reference the DB IDs of the deleted Business Attributes.

Multi-Tenancy for GMS/CS

In order to support multi-tenancy and business units, GMS/CS now supports additional HTTP headers:

- `ContactCenterId` contains the Tenant DBID or the Tenant Name.
- `GroupId` contains optional business unit name.

You can use these headers to specify the tenant and business unit for a given request. If your request does not include these headers, the Context Services handles the request in single-tenant mode and uses the provisioned tenant ID with no business unit name of your GMS/CS application.

If you provide an incorrect `ContactCenterId` string, your application receives the following error message:

```
{
    "message": "Invalid tenant specified : no tenant specified in multi-tenant environment.
Verify configuration and 'organization' HTTP header.",
    "exception":
"com.genesyslab.gsg.services.contextservices.ContextServicesExceptionResource"
}
```

> ## Important
>
> You must make sure that you provide these IDs in HTTP headers, not as part of the JSON Body content.

Read Also

- Profiles and Identification
- Services, States, and Tasks

Basic Access Authentication

This page offers guidelines for managing Authentication with the Context Services.

About Basic Authentication

Wikipedia Basic Access Authentication states that:

In the context of an HTTP transaction, the basic access authentication is a method designed to allow a web browser, or other client program, to provide credentials – in the form of a user name and password – when making a request.

The Context Services provides support for basic access authentication once enabled in the authentication section of your configuration.

- If basic access authentication is enabled, the REST requests must contain a valid username and password in the HTTP/HTTPS header. As a result, the Context Services sends descriptive error messages if it receives an incorrect username/password combination.

- If basic authentication is disabled, the Context Services ignores any username or password passed in HTTP/HTTPS header.

If the authentication is enabled and valid information is not provided, the Context Services returns the HTTP response 401 Unauthorized. In that case, you should resubmit the request with the proper authentication header.

Base64 Encoding

The authentication string to transmit is the result of the concatenation of the username and password separated by a colon (*username:password*). It must then be encoded with the Base64 algorithm. For example, if the username is 'kent' and the password 'superman', the string to encode is kent:superman and results in the string 'a2VudDpzdXBlcm1hbg=='.

If you are using a framework, it may provide the Base64-encoding transparently. If your framework does not include the Base64-encoding feature then you must encode your string. The following code snippet shows how to proceed with a Restlet application:

```
final Request request = new Request();
String url = "http://" + host + ":" + port + "/server/status";
request.setResourceRef(url);
request.setMethod(Method.GET);
final Client myClient = new Client(Protocol.HTTP);
ChallengeResponse authentication = new ChallengeResponse(ChallengeScheme.HTTP_BASIC, "kent",
"superman");
request.setChallengeResponse(credential);
Response response = client.handle(request);
```

> **Important**
>
> Additional examples of Base64 encoding are available in Wikipedia Basic Access Authentication.

Request Flow and Returned Errors

The following sequence diagrams show the protocol request and answer flow when basic access authentication is enabled.
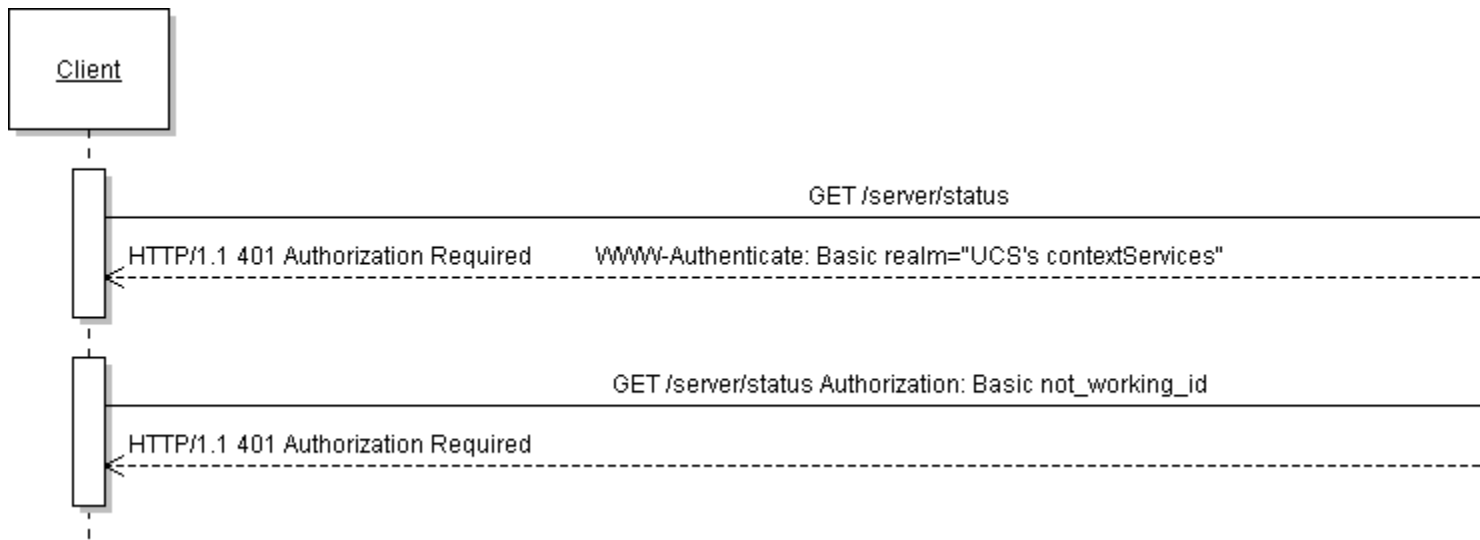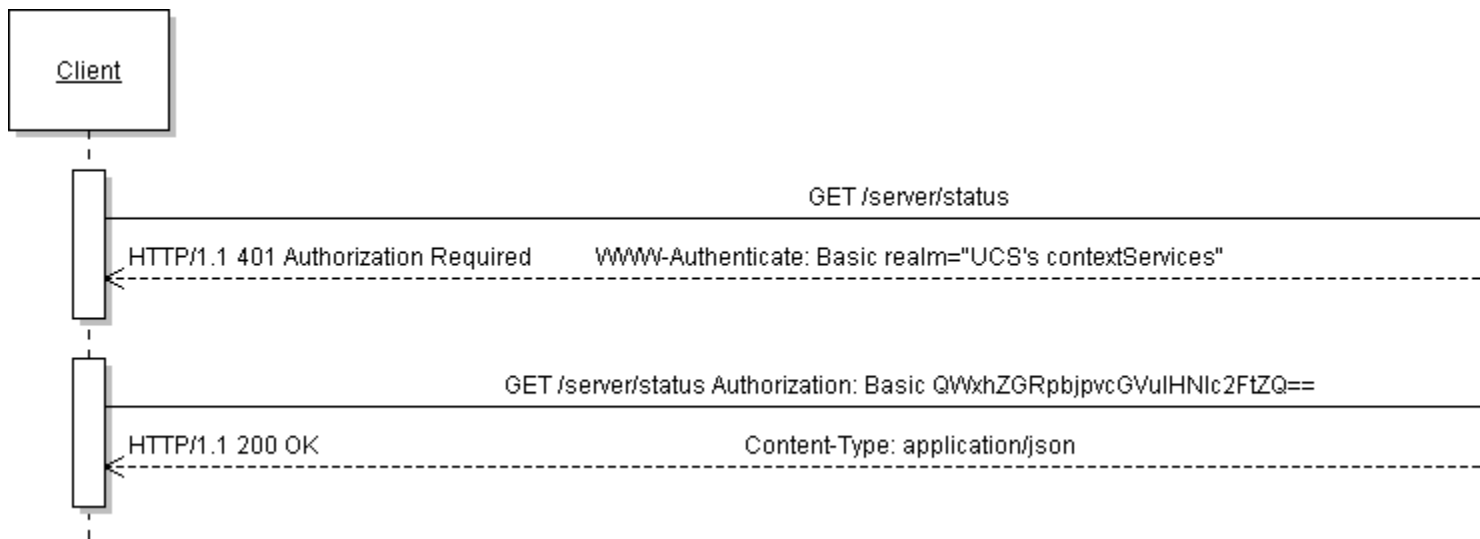
If the request returns the 401 Unauthorized error, your application should retry with a correct HTTP header. The Context Services returns 401 Unauthorized error due to authentication issues in the following scenarios:

- The authentication is enabled and the request is not authorized.

- The request provides the correct header for authentication, but wrong credential information (the username or the password is wrong).

Role-Based Access Control

This page describes how you can implement the role-based access in the Context Services.

Configuration

Through Configuration Manager or Genesys Administrator, you can define roles for your application built on top of the Context Services. To do this, you assign one or more roles to your users when creating your application's configuration in the Context Services. You are responsible for creating and defining these roles, where each role is a collection of Genesys Administrator Tasks associated with permissions.

<tabber> Context Services=

Rights for Context Services

Tasks related to Service management are available in GMS and may require specific permission set up in Genesys Administrator.

Starting 8.5.0, privileges are simplified.

Name	Description
Administrator	Specifies write access to all CS APIs.
Supervisor	Specifies read access to all CS APIs.

The following table details the relationship between requests and privileges.

Privileges required per API operations:

HTTP Operation	Required Permissions
PUT	Administrator
POST	Administrator
GET	Administrator or Supervisor
DELETE	Administrator

Click here to learn how to create roles and assign privileges.

|-| Rights for Customer Profiles=

Genesys Administrator Tasks for Customer Profiles in UCS

Tasks related to Customer management are available in UCS and require specific permission set up in Genesys Administrator.
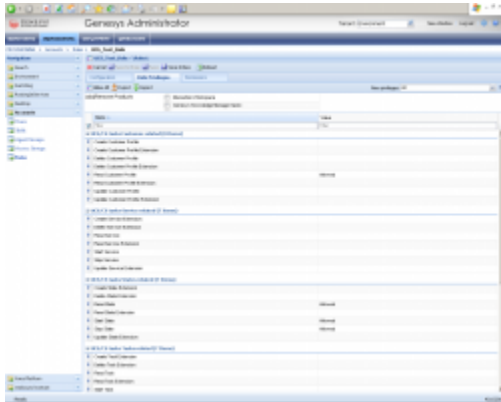
Once authenticated, if the use-role option is set to true in the configuration (see the options defined in [authentication] Section) then the Universal Contact Server checks that each operation is allowed. If not, Error 403 forbidden is returned.
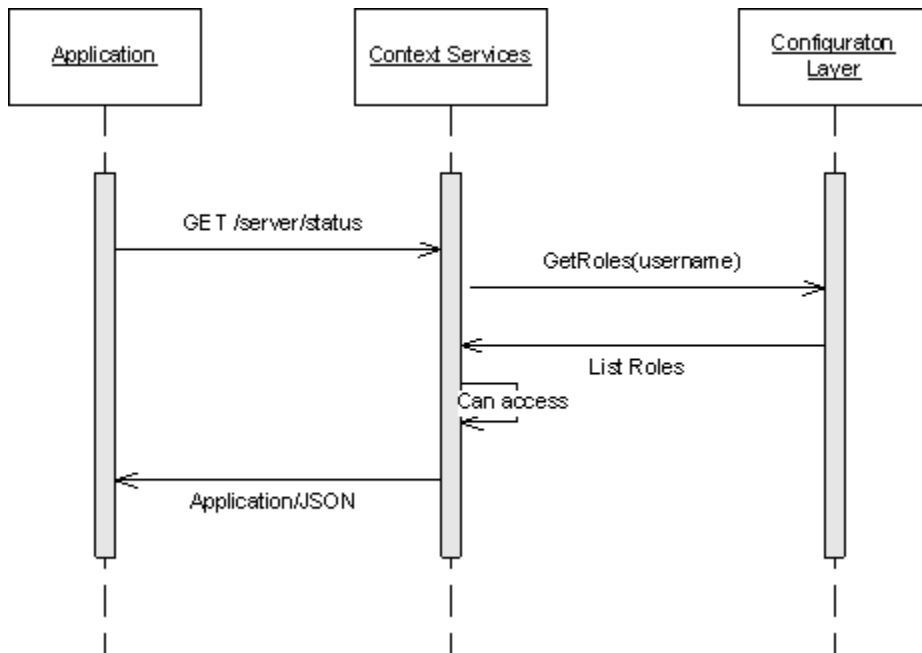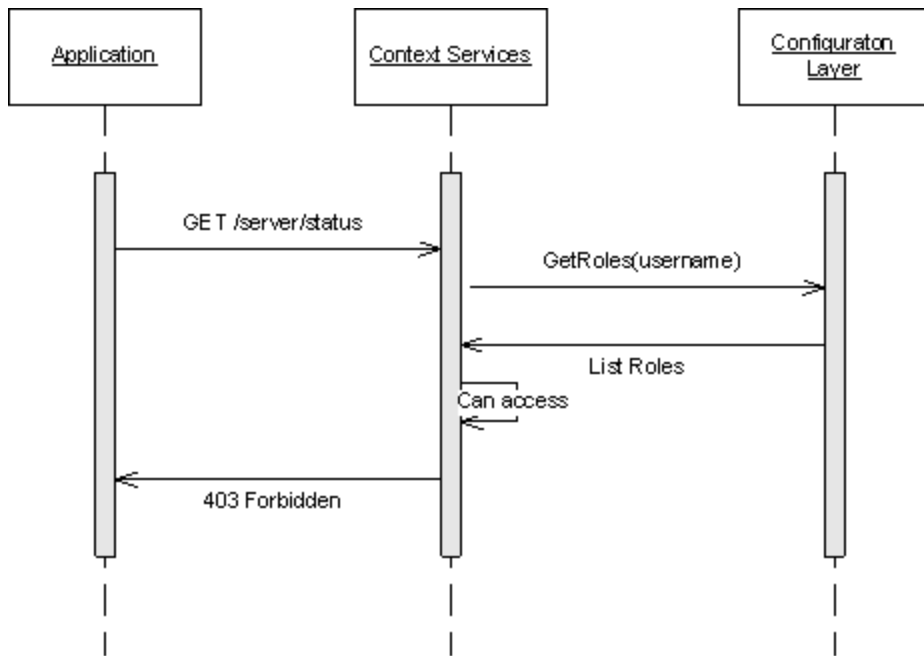
Mapping Genesys Administrator Task with Context Services

Operations can require that one or more Genesys Administrator Tasks are allowed, according to the type of data that the request modifies. If your application's role does not allow all of the rights required for a given operation, then the operation does not proceed.

For example, consider that your application performs a Create Customer Profile operation with extensions. If your application's role allows UCS.Customer.createProfile but not UCS.Customer.createProfileExtension then the profile is not created. Your application instead receives a HTTP 403 Forbidden error.

Operation	Genesys Administrator Tasks
Profile Operations	
Create Customer Profile POST /profiles	• UCS.Customer.createProfile • UCS.Customer.createProfileExtension (if extensions)
Delete Customer Profile DELETE /profiles/${customer_id}	• UCS.Customer.deleteCustomerProfile

Operation	Genesys Administrator Tasks
Delete Record From Profile Extension PUT /profiles/${customer_id}/extensions/${ext_name}/by/unique	• UCS.Customer.deleteProfileExtension
Identify Customer GET /profiles	• UCS.Customer.readCustomerProfile • UCS.Customer.readProfileExtension (if `include_extensions` is specified in the query)
Insert Extension Records POST /profiles/${customer_id}/extensions	• UCS.Customer.createProfileExtension
Bulk Profile Import POST /profiles/import	• UCS.Customer.executeBulkImport • UCS.Customer.createProfile • UCS.Customer.createProfileExtension
Query Customer Profile GET /profiles/${customer_id}	• UCS.Customer.readCustomerProfile • UCS.Customer.readProfileExtension (if extensions)
Update Customer Profile PUT /profiles/${customer_id}	• UCS.Customer.updateCustomerProfile • UCS.Customer.updateProfileExtension (if extensions)
Merge Customer Profile PUT /profiles/${customer_id}/merge/${src_id}/	• UCS.Customer.mergeCustomerProfile
Update Record In Profile Extension PUT /profiles/${customer_id}/extensions/${ext_name}/by/unique	• UCS.Customer.updateProfileExtension
Schema Operations	
Create Profile Extension Schema POST /metadata/profiles/extensions	• UCS.SchemaMgt.createProfileExtensionSchema
Create Identification Key POST /metadata/identification-keys	• UCS.SchemaMgt.createIdKeys
Get Identification Keys GET /metadata/identification-keys	• UCS.SchemaMgt.readIdKeys
Query Profile Schema GET /metadata/profiles/	• UCS.SchemaMgt.readProfileExtensionSchema
Query Profile Extension Schema	UCS.SchemaMgt.readProfileExtensionSchema

Operation	Genesys Administrator Tasks
GET /metadata/profiles/extensions	
Query Business Attribute Schema GET /metadata/business-attributes/${business-attribute-name}	• UCS.SchemaMgt.readBusinessAttributes
Get Metadata Cache GET /metadata/cache	• UCS.SchemaMgt.handleMetadata
Change Metadata Cache PUT /metadata/cache	• UCS.SchemaMgt.handleMetadata
Get Metadata GET ${contenttype}} /metadata	• UCS.SchemaMgt.handleMetadata
Delete Metadata Profile Extensions DELETE /metadata/profiles/extensions/${extension-name}	• UCS.SchemaMgt.deleteProfileExtensionSchema
Delete Metadata Identification Keys DELETE /metadata/identification-keys/${id_key-name}	• UCS.SchemaMgt.deleteIdKeys
Interaction Operations	
Query Interactions GET /customers/${customer_id}/interactions GET /services/${service_id}/interactions GET /interactions/${interaction_id}	• UCS.SchemaMgt.readInteraction

Export Features
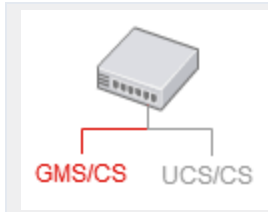
This page details the export features available for GMS/CS services.

Introduction

You can export services data by performing one of the following queries:

- Export the services stream to JSON
- Export the services stream to JSON or CSV files

These two export features will select the exported services for a given time-range, based on given `time_from` and `time_to` parameters.

Important

Only services which received events during the specified time-range can be exported.

By default, any services which received a start or completed event during the given period are exported, regardless their current state.

Setting Filters

If you set the state filter, the response filters the services according to their current status, active or not, and regardless the given time range.
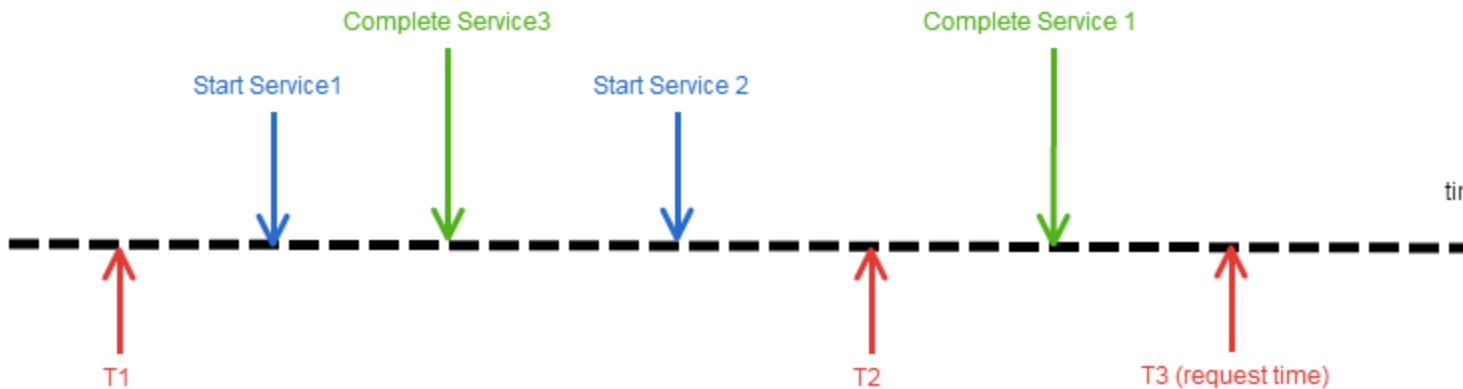
Exporting services with event occurrences between T1 and T2. T3 is the date of the query.

The table below provide examples of filter values and their results according to the above figure.

Filter examples

Filter - filter_events	Filter - filter_state	Details	Result
filter_events = any	filter_state = any	The response includes the services which received events between T1 and T2.	S1, S2, S3
filter_events = any	filter_state = active	The response includes the services which received events between T1 and T2, AND which are still active at T3.	S2
filter_events = any	filter_state = inactive	The response includes the services which received events between T1 and T2, AND which are no longer active at T3:	S1, S3
filter_events = started	filter_state = any	The response includes the services which received a start event between T1 and T2.	S1, S2
filter_events = started	filter_state = active	The response includes the services which received a start event between T1 and T2, AND which are still active at T3.	S2
filter_events = started	filter_state = inactive	The response includes the services which received a start event between T1 and T2,	S1

Filter - filter_events	Filter - filter_state	Details	Result
		AND which are no longer active at T3.	
filter_events = completed	filter_state = any	The response includes the services which received a end event between T1 and T2	S3
filter_events = completed	filter_state = active	The response includes the services which received a end event between T1 and T2, AND which are still active at T3. **Tip** This combination can not return results. You should not use it.	N/A
filter_events = completed	filter_state = inactive	The response includes the services which received a end event between T1 and T2, AND which are no longer active at T3.	S3

Exporting to the JSON Stream

If you export the services stream to JSON, you can either export a list of IDs or the whole data. Be careful: exporting the whole data can consume a lot of resources and delay the response. This query is intended to be used to retrieve a list of IDs, and you should then query the service data based on the service ID.

If you think that your query may retrieve a long list of IDs and if you need the service data, you may export the service stream to JSON et CSV files instead of setting the `export_content` option to `true`.

> **Important**
>
> If you set the `export_content` option to `true`, the response time depends on the size of the service data and the number of retrieved services.

Exporting to the Files Stream

If you need to export a wide collection of service data, you should export the services stream to CSV or JSON files. In this case, you can still use filters to select the exported content and you should not face timeout issues.

> **Important**
>
> The size of the generated files depend on the size of the service data.

Customer Profile API

> **Important**
>
> **Prerequisites:** You need to enable profiles in UCS.

This page provides guidelines for managing Customer profiles and interaction information. In 8.5, UCS is still responsible for the management of customer-related information. Queries and resources did not change since 8.1.

Introduction

The Customer Profiles API includes all the information stored in the Universal Contact Server:

- Customer profile (contact information)
- Interactions
- Schemas

> **Important**
>
> You don't need Customer Profiles to run the Context Services, as detailed in the Architecture section. This feature did not evolve since 8.1 and does not include some of the new features available for the GMS/CS part of the Context Services. Check the migration page for further details.

Going Further

- Learn about profile and identification
- Learn about groups of profiles
- Learn about profile extensions and schemas

Profiles and Identification

> **Important**
>
> **Prerequisites:** You need to enable profiles in UCS.

This page gives guidelines for managing Customer Profiles with Context Services. All customer resources and REST requests are redirected to UCS.

Learn about the Customer Profile and Associated Resources

The Customer Profile resource associates a customer ID with:

- A list of attributes, built on top of the existing UCS Contact model.

- A list of extensions, defined at runtime through Context Services

The attributes correspond to the core information defined in Universal Contact Server. Their schema are defined through the list of Business Contact Attributes that you can define in the Configuration Manager (see Configuration Options for additional details). The extensions are additional information that your application can create so that you can extend the profile at runtime, as explained in Extending the Customer Profile. Identification Keys define which attributes should be used to identify a customer in the database. For instance, the association of [LastName, Firstname], or the email address.

Attribute Values

Attributes and extension records can be either:

- "single-valued"(for instance, LastName, FirstName, identifiers, and so on);

- "multi-valued": values can be multiple (for instance, phone numbers, email addresses, and so on).

The following output presents a sample of Customer profile, where "FirstName", "LastName", and "DOB" (Date Of Birth) are single-valued contact attributes, and the other fields are multi-valued extension records created at runtime.

```
{
    "FirstName": "Bruce",
    "LastName": "Banner",
    "DOB": "1962-05-10",
    "EmailAddress": [
        "bruce.banner@marvelous.com",
        "b.banner@hulk.dom"
    ],
    "Phone": [
```

```
{
    "PhoneType":0,
    "prefix":"+33",
    "PhoneNumber":"3145926535",
    "description":"family phone",
    "start_availabilty":"2009-12-18T18:30:00.000Z",
    "end_availabilty":"2009-12-18T21:40:00.000Z"
},
{
    "PhoneType":2,
    "prefix":"+33",
    "PhoneNumber":"6543210",
    "description":"business calls only, no sales",
    "start_availabilty":"2009-12-18T09:30:00.000Z",
    "end_availabilty":"2009-12-18T17:45:00.000Z"
},
{
    "PhoneType":5,
    "prefix":"+33",
    "PhoneNumber":"951357456",
    "description":""
}
]
}
```

Profile Content

The content of the Customer Profile follows a schema (a translation of the Business Contact Attributes), which describes its content with a list of Attribute Schema, apart from extension content, as shown in the following output example:

```
{"encrypt":false,"name":"PIN","length":256,"type":"string"},
{"encrypt":false,"name":"Title","length":256,"type":"string"},
{"encrypt":false,"name":"CustomerSegment","length":256,"type":"string"},
{"encrypt":false,"name":"LastName","length":256,"type":"string"},
{"encrypt":false,"name":"AccountNumber","length":256,"type":"string"},
{"encrypt":false,"name":"FirstName","length":256,"type":"string"},
{"encrypt":false,"name":"PhoneNumber","length":256,"type":"string"},
{"encrypt":false,"name":"ContactId","length":256,"type":"string"},
{"encrypt":false,"name":"EmailAddress","length":256,"type":"string"},
```

> ### Important
> The profile schema does not contain information related to extensions.

At runtime, your application can retrieve this schema through the Query Profile Schema operation. Your application cannot modify the profile schema through Context Services:

- If you wish to modify the profile schema, make modifications to in Configuration Manager via the Business Attribute "ContactAttributes".

- If you wish to add additional information to the profile at runtime, use the profile extensions.

Profile Extensions

Extensions related to customer profiles used to be described as extensions in former versions and they are now detailed on the Profile Extension page of the API reference. They are not flexible JSON data as extensions used in the services handled on the GMS side.

Extensions allow you to extend the content of a profile with additional records. Your application is fully responsible for the creation and management of this optional content. As stated in the Preliminary Steps to Using Profiles section, your application must define the extensions' content by creating a schema for the extension.

Once the schema is defined, your application can use the extensions by specifying extension information to the profile submitted with the operations' body, or by using dedicated operations, as presented below.

Identification Key

The identification key is a combination of attributes used to identify a customer. These attributes (one or more) belong to the customer profile and/or its extensions.

> ### Important
> Identification keys cannot be issued from both the profile and its extensions.

Some identification key samples:

- An identification key consisting of the "name" and "birthdate" attributes from the profile.
- A key based on the "pin" attribute of a single-valued extension used to hold customer pass codes.
- A key based on the "number" attribute of a multi-valued extension used to record the phone numbers we have seen a customer call from.

When UCS receives a new customer profile and the associated extension data, it builds the indexing structures according to the specified identification keys, in order to ensure efficient customer identification.

> ### Important
> Read Identification Key for a resource example.

Preliminary Steps to Using Profiles

Before your application can start interacting with customers and recording their information by using Customer Profile resources, you must create certain types of information that enable you to manage

these profiles. This includes the profile schema, extension schemas, and identification keys.

Creating Extension Schema (Optional)

If your application only needs to use the predefined profile attributes, you can use them as provided. But if you need to work with information that is not contained in the existing attributes -- for instance, for use as identification keys -- you can create extensions that contain this extra information by creating an extension schema for each additional piece of information.

> ### Important
> Your application cannot use an extension if the associated schema does not exist.

As for the profile schema, the profile extension schema is composed of Attribute Schema which define the schema content. The following output shows the definition of the single-valued "Address" extension.

```
"name":"Address",
  "type":"single-valued",
  "attributes": [
       {"name":"AddressType","type":"integer","default":0},
       {"name":"Address","type":"string","length":256},
       {"name":"City","type":"string","length":32},
       {"name":"County","type":"string","length":32},
       {"name":"PostCode","type":"string", "length":10},
       {"name":"Country","type":"string","length":32}
       ]
```

The creation of the profile extension is possible through the Create Profile Extension Schema operation. See this example section for detailed information.

Creating Identification Keys (Mandatory)

Operations and resources cited in this section are part of Schema Operations

If your application does not specify identification keys, the only way to access customer information is to retrieve the customer profile with the customer ID. However, you can use the Create Identification Key operation to set up an Identification Key. You create these keys by using profile attributes. For example,you can use a single attribute, or a combination of attributes ("LastName" and "FirstName" for instance), or a combination of extension attributes if you have created a schema for a profile extension.

You can define as many identification keys as you need, but you should consider that creating too many identification keys will slow down creation, update, and removal operations. Once identification keys have been created, your application can continue to work with customer data, accessing it by means of the Get Identification Keys operation.

> **Important**
>
> Your application only needs to register these schemas and identification keys once.

Managing Profiles: Creation, Identification, Extensions

The Customer Profile and its extensions should only be created for the storage of customer-level information. You should use the results of interactions and dialogs with the customer to fill in service, state, and task information.

Managing a Profile

The Create Customer Profile operation creates an entry for the Customer Profile in UCS Database. This step must be completed prior to using the profile data in other operations such as:

- Query Customer Profile
- Update Customer Profile

Identify a Customer

Operations and resources in this section are part of Schema Operations.

The operation Identify Customer enables your application to retrieve customer profiles based on a few attribute values passed in as parameters, without specifying the customer ID, as shown in this example: GET /profiles/
contacts.phone_number=408-888-3214&extensions=contacts,purchases€clude_profile=yes
€clude_extensions=unique If no customer profile is returned, your application can create a new profile based on the current set of information available.

This lets your application to determine the customer's identity without having to gather as much information. For instance, if your application deals with calls by using a customer phone number, the customer is easy to identify if he or she calls back.

Extending the Customer Profile

As stated in the introduction sections, your application can add new types of information to the basic customer profile by using extensions. If your application needs to record a specific set of data (for instance, email addresses), your application can create a schema for this extension. Once an extension schema has been created in UCS, your application can use the new extension and create associated records. Your application can either:

1. Insert Extension Records for a given customer, or
2. Create Customer Profile or Update Customer Profile with extension records.

Importing Customer Profiles

The Bulk Profile Import operation enables to import a `.csv` file which contains a wide set of contacts. The `.csv` file must be compliant with RFC4180 and already available on the UCS local file system. Profiles must match metadata and can include extensions. An identification key can be set to avoid ambiguities with former profiles when updating the profile database. See Bulk Profile Import for examples and details.

References

Grouping Customer Profiles

> **Important**
>
> **Prerequisites:** You need to enable profiles in UCS.

This page helps you to learn how you can group customer profiles.

Overview

In the following example, the requirement is to enable an "account" or "family" view of multiple customer profiles. If a new object is created, the profile entity and attribute `Group` is set to `true`. If a multi-valued profile extension points to a group member profile, id-keys can be used. It then becomes possible to either attach services to the group, or to the member depending on the scenario.

Examples

The following examples are similar with the difference being the way the profiles are created. Relationship between profiles and group are either "direct link" from entity account owner to other profiles, or "indirect link" where each profile belongs to a family profile.

Account

There are many Profiles. Some are **Admin** for one or more other Profiles. For example, a telephony provider has a single account (billing account) for several persons. One of the persons is an *admin* for the account and has rights to change options for each cellular.

Example data:

- Profile with Id=**"XXXXXXXXXXX-JOHN":**

```
{
 CustomerId="XXXXXXXXXXX-JOHN",
 LastName:"Doe",
 FirstName:"John",
 Cellular:"555-123456",
 EmailAddress:"john@doe.net",
 GroupAdmin: [
   {AdminForProfile:"XXXXXXXXXXX-JANE"},
   {AdminForProfile:"XXXXXXXXXXX-PETER"}
 ],
 ProviderOptions: { AccountInfo:"account number", MemberLevel:"High" }
}
```

- Profile with Id=**"XXXXXXXXXXX-JANE":**

```
{
 CustomerId="XXXXXXXXXXX-JANE",
 LastName:"Doe",
 FirstName:"Jane",
 Cellular:"555-987654",
 EmailAddress:"jane@doe.net",
 ProviderOptions: { AccountInfo:"account number", MemberLevel:"Untouchable" }
}
```

- Profile with Id=**"XXXXXXXXXX-PETER":**

```
{
 CustomerId="XXXXXXXXXXX-PETER",
 LastName:"Doe",
 FirstName:"Peter",
 Cellular:"555-654321",
 EmailAddress:"peter@doe.net",
 ProviderOptions: { AccountInfo:"account number", MemberLevel:"Untouchable" }
}
```

- Two extensions for the example:

  - Single-valued extension **ProviderOptions** with any type of attributes used for the example to customize Cellular options.

  - Multi-valued extension **GroupAdmin** with attribute **AdminForProfile**

  - Identification keys :

    - on attribute **EmailAddress** from Core Profile

    - on attribute **Cellular** from Core Profile

    - on attribute **LastName+FirstName** from Core Profile

    - on attribute **AdminForProfile** of extension **GroupAdmin**

- Example scenario:

  - John calls to update his cellular account.

  - He is identified by his Cellular="555-123456".

  - The system knows that he his and admin for Jane and Peter because of the **GroupAdmin** extension.

  - He is asked "Do you want to change settings for your account #1, for Jane's account #2 or for Peter's account #3?".

  - If he enters #2 or #3, the system picks the correct Profile by the Id and can retrieve specific cellular options.

  - ...

  - Peter calls to change some options.

  - He is identified by Cellular="555-654321".

  - The system knows he is not Admin.

  - Depending on the requests, the system may "identify" who is admin for the cellular by the id-key on **GroupAdmin.AdminForProfile="XXXXXXXXXX-PETER"**.

- The system might fail the request stating "need admin rights".

Family

Each member of a family has its own profile. They belong to the Family Group (same house hold). This result in slightly different from the previous example because the Family itself is identified as a profile.
Note: The Account example can also be implemented this way.

Example data:

- Profile with Id=**"XXXXXXXXXXX-JOHN"**:

```
{
 CustomerId="XXXXXXXXXXX-JOHN",
 LastName:"Doe",
 FirstName:"John",
 Cellular:"555-123456",
 EmailAddress:"john@doe.net",
 GroupFamily: { ProfileFamily:"XXXXXXXXXXX-DOE" }
}
```

- Profile with Id=**"XXXXXXXXXXX-JANE"**:

```
{
 CustomerId="XXXXXXXXXXX-JANE",
 LastName:"Doe",
 FirstName:"Jane",
 Cellular:"555-987654",
 EmailAddress:"jane@doe.net",
 GroupFamily: { ProfileFamily:"XXXXXXXXXXX-DOE" }
}
```

- **Family = Profile 'XXXXXXXXXXXX-DOE'**:

```
{
 CustomerId="XXXXXXXXXXXX-DOE",
 LastName:"Doe",
 PhoneNumber:"555-1592648",
 EmailAddress:"family@doe.net",
 PostalAddress: { Address:"5, This Road", ZipCode:65536 }
}
```

- Extensions:

  - Single-valued extension **PostalAddress** with attributes like 'Zip Code', 'State', etc. This extension may have values only for the **Family** since all profiles are to live at the same place.

  - Single-valued extension **GroupFamily** with attribute **ProfileFamily** pointing to the main family profile.

- Identification keys:

  - on attribute **EmailAddress** from Core Profile.

  - on attribute **Cellular** from Core Profile.

  - on attribute **PhoneNumber** from Core Profile.

- on attributes **LastName+FirstName** from Core Profile.

- on attribute **ProfileFamily** from extension **GroupFamily**.

- Example scenario:

  - Assuming John sends the request from e-mail or cellular:

    - He is identified by id-key **Profile.Cellular="555-123456"** or **Profile.EmailAddress= "john@doe.net"**.

    - Then his family information is gathered from querying Profile with Id **GroupFamily.ProfileFamily="XXXXXXXXXXXX-DOE"**.

  - Assuming John calls from Home:

    - His **Family** information is matched by id-key **Profile.PhoneNumber="555-1592648"**.

    - Members of the family can be identified by id-key on **GroupFamily.ProfileFamily= "XXXXXXXXXXXX-DOE"**.

    - The IVR might question "Who are you? Jane or John?".

Profile Extensions

> **Important**
> **Prerequisites:** You need to enable profiles in UCS.

This page provides guidelines for managing Profile Extensions in UCS.

About Profile Extensions

Profile extensions are stored in the UCS database and are not JSON values as service extensions. Their management is similar to former extension management in Context Services. You must define database schemas before you can use them in your application.

Profile extensions are additional information which extend the standard contents of resources such as Customer Profile. A Extension is a record-a list of attributes-or an array of records, associated with a resource ID.

- You can define as many extension types as you need by creating an Extension Schema for each of them.

- Extension schema are created through Context Services (see List of Schema Operations), not through the Configuration Layer (Configuration Manager).

Extension records can be either:

- "single-valued": The extension contains a single record across the resource (for instance, LastName, FirstName, identifiers, etc.)

- "multi-valued": The extension can contain several values (for instance, phone numbers, e-mail addresses, etc.)

Extensions are provided at the same time and at the same level than the attributes of the resource. For instance, the following output presents a profile containing the attributes FirstName, LastName, DOB<ref>DOB: Date Of Birth</ref> and one multi-valued extension *EmailAddress*:

```
{
    "FirstName": "Bruce",
    "LastName": "Banner",
    "DOB": "1962-05-10",
    "EmailAddress": [
        "bruce.banner@marvelous.com",
        "b.banner@hulk.dom"
      ]
}
```

Unique Attributes

In the case of multi-valued extensions, the attributes which are part of the 'unique' list (specified in the Extension Schema) are used to identify records. The combination of these attributes' values must be unique across the related resource, and this enables UCS to identify a given record in the given extension. For example, consider a 'Bill' extension which includes the attribute *bill_id*. To ensure that a given service does not have two 'Bill' extensions with the same *bill_id*, set the following unique array in the extension schema:

```
unique = ["bill_id"]
```

The attributes of the unique list are mandatory at the extension record's creation. You need to provide values for the 'unique' attributes:

- At the creation of an extension record.
- In operations which update or delete a specific record, such as Update Record In Profile Extension or Delete Record From Profile Extension.

> ## Warning
>
> Operations which manage extension records are part of the related resource operations. For instance, the operations which manage records of profile extensions are part of the List of Profile Operations.

Limitations

- Once created, you cannot update the schema.
- When you are dealing with extensions or extension schema, make sure that you do not use one of the

Unauthorized Strings as an attribute name or value.

Managing Extension Schema

Operations and resources in this section are part of

Before you can start using extensions, you must create their schema.

> ## Important
> Once created, you cannot update or remove them.

You can create schema with the following operations:

- Create Profile Extension Schema

Then, you can retrieve extension schema.

- Query Profile Schema
- Query Profile Extension Schema

Example: Retrieving the schema for profile extensions:

```
GET /metadata/profiles/extensions
```

Result

```
200 OK
[
{
    "name":"Phone",
    "type":"multi-valued",
    "attributes": [
        {"name":"PhoneType","type":"integer","default":0,"mandatory":"true"},
        {"name":"prefix","type":"string","length":"3","default":"555",},
        {"name":"PhoneNumber","type":"integer","length":15,"mandatory":"true"},
        {"name":"description","type":"string","length":32,"mandatory":"true"},
        {"name":"start_availabilty","type":"datetime"},
        {"name":"end_availabilty","type":"datetime", "mandatory":"false"}
        ]
},
{
    "name":"Address",
    "type":"single-valued",
    "attributes": [
        {"name":"AddressType","type":"integer","default":0},
        {"name":"Address","type":"string","length":256},
        {"name":"City","type":"string","length":32},
        {"name":"County","type":"string","length":32},
        {"name":"PostCode","type":"string", "length":10},
        {"name":"Country","type":"string","length":32}
        ]
} ]
```

Managing Extensions

Adding Extensions to a given Resource

You can add extensions when managing the resources with related operations which authorize the <extension n> attribute in the operation's body. In that case, if a former value of the extension exists for the given resource, this former extension value is replaced with the new extension value specified in the body.

Let's consider the following multi-valued extension record named 'Satisfaction'. The unique field which identifies records is "place" (the name of the proposed place for the booking.

Example: Records for a 'Satisfaction' extension

```
PUT /profiles/00027a52JCGY000M
  {
 "FirstName": "Bruce",
 "LastName": "Banner",
 "DOB": "1962-05-10",
 "EmailAddress": [
       "bruce.banner@marvelous.com",
       "b.banner@hulk.dom"
 ],
 "Address": { "Type":1, "Address":"21 JumpStreet", "City":"Hollywood",
              "County":"Santa Barbara", "PostCode":"555", "Country":"United States" }
}
POST /profiles/00027a52JCGY000M/extensions
  {
"customer_id":"00027a52JCGY000M",
"Satisfaction": [
   {
  "rating":2,
  "pertinence":8,
  "usefull":true,
  "place":"Terranova mexico resort"
 },
 {
  "rating":8,
  "pertinence":4,
  "usefull":false,
  "place":"Fancy resort Paris"
 }
]
```

Example: Operation which updates the 'Satisfaction' extension

```
POST /profiles/00027a52JCGY000M/extensions
  {
"customer_id":"00027a52JCGY000M",
  "Feedback":
  {
    "FeedbackType":"survey",
    "rating":7,
    "notes":"warm welcome at frontdesk, thanks for the nice trip"
  },
  "Satisfaction": [
  {
   "rating":2,
   "pertinence":6,
   "usefull":true,
   "place":"Marina Porto Vecchio"
  }
  ]
 }
```

As a result, the previous records 'Fancy resort Paris ' and 'Terranova mexico resort ' are lost. In this case, to add a new record to the extension, you must specify the whole extension content. For instance, note the following:

Example: Operation which updates the 'Satisfaction' extension without losing records

```
POST /profiles/00027a52JCGY000M/extensions
  {
"customer_id":"00027a52JCGY000M",
  "Feedback":
  {
    "FeedbackType":"survey",
    "rating":7,
    "notes":"warm welcome at frontdesk, thanks for the nice trip"
   },
  "Satisfaction": [
      {
       "rating":2,
       "pertinence":6,
       "usefull":true,
       "place":"Marina Porto Vecchio"
      },
    {
       "rating":2,
       "pertinence":8,
       "usefull":true,
       "place":"Terranova mexico resort"
    },
    {
       "rating":8,
       "pertinence":4,
       "usefull":false,
       "place":"Fancy resort Paris"
    }]
 }
```

Retrieving Extensions

GET operations which enable to retrieve resources include the "extensions" parameter to specify a list of extensions to retrieve. By default, extensions are not returned. The following list is not exhaustive:

- Query Customer Profile

Deleting an Extension

To delete the extension of a given resource, use the related *Update XXX Extension* operation with no attributes in the operation's body.

- Update Customer Profile and Update Record In Profile Extension

Example: Deleting the *relatedOffers* multi-valued extension of the customer *00027a52JCGY000M*

```
PUT
POST /profiles/00027a52JCGY000M/extensions

 []
```

As explained in Role-Based Access Control, you need update privileges to clear the extension, as follows:

- Clear Profile Extension—UCS.Customer.updateProfileExtension

Read More

- Profiles and Identification

Customizing Journey Timeline

> **Important**
>
> **Prerequisites:** You need to enable profiles in UCS.

Customizing Journey Timelines

All customization can be done in a few steps:

1. Edit the `<Context_Services_HOME>/files/configprofileconfig.json` file that contains all information relative to the displayed data of the timeline UI.

2. Read further sections explaining how you can create new data to be displayed for both states and profiles, then edit the `profileconfig.json` file.

3. Be sure to restart your Context Services application after you saved your edits.

> **Important**
>
> Make sure to save this file in UTF-8.If you remove the `profileconfig.json` file, then Journey Timeline loads the default settings.

Customizing Profiles

You can edit the information displayed in the GLOBAL_PROFILE_EXTENSIONS section of the profileconfig.json file:

- Add or remove information in `mapping` and `CustomerProperties` section.

- Edit icons and color for `customerName`, `EmailAddress`, and `PhoneNumber` fields.

- Add or remove search menu items in the searchOptions section.

```
"GLOBAL_PROFILE_EXTENSIONS":{
    "mapping":{"Phone Number": "PhoneNumber", "Email":"EmailAddress", "Customer
ID":"customer_id", "Anonymous ID": "Anonymous ID", "Country": "Country", "AddressState":
"State", "Title":"Title", "City":"City", "Language":"Language"},
    "searchOptions":{"Phone Number": "PhoneNumber", "Email":"EmailAddress", "Customer
ID":"customer_id", "Anonymous ID": "Anonymous ID"},

    "ExtraQueryOptions": {"include_profile":"yes"},

    "CustomerProperties": [{"label":"customer_id", "value":"Customer ID", "showInTable":"no",
"primaryKey":"yes"},
                          {"label":"LastName", "value":"Last Name"},
                          {"label":"FirstName", "value":"First Name"},
                          {"label":"PhoneNumber", "value":"Phone"},
```

```
                            {"label":"EmailAddress", "value":"Email"},
                            {"label":"Country", "value":"Country"},
                            {"label":"AddressState", "value":"State"},
                            {"label":"Title", "value":"Title"},
                            {"label":"City", "value":"City"},
                            {"label":"Language", "value":"Language"}],


    "customerName": {"icon":"fonticon icon-person", "color": "#2E69DB"},
    "EmailAddress": {"icon":"fonticon icon-email", "color":"#555D66"},
    "PhoneNumber": {"icon":"fonticon icon-phone", "color":"#A6AAAF"}
  }
```

Reversing Timeline Display

Introduced in: 8.5.207.05

The Journey Timeline can display a Conversation in reverse order with the latest events displayed on top of history. To activate the reverse order, edit the PROFILE_CONFIG section of the profileconfig.json file and add the following attribute:

```
"TIMELINE_CUSTOMIZATION": {
    "stateOrder": "reverse"
}
```

Customizing Display for States
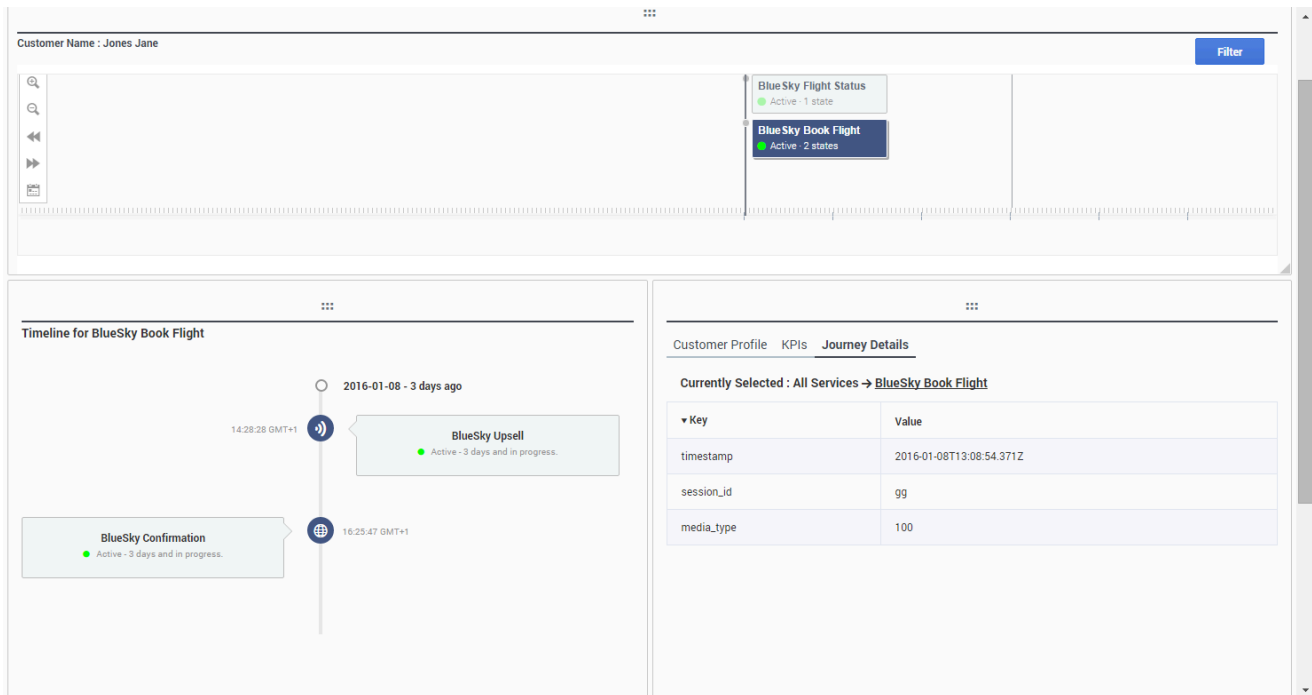
The UI displays name and state information about states in the timelines and details panels. You can expand this information by configuring which JSON extension your wish to display in the "STATE_EXPRESSIONS" section of the `profileconfig.json` file as explained below.

Mapping with extension

To add a JSON extension, to a service, state, or task, all you need to do is to specify the JSON extension value when creating or updating your resources with a REST query. For instance, you can define a new **CRMData** state extension with a simple POST query at state completion:

```
POST http://localhost:8080/genesys/1/cs/services/711982/states/5362/end
{
        "service_id": 711982,
        "CRMData": {
            "caseNumber": "GF4GHL",
            "caseOwner": "Bogota 11:00 - 12:00"
        },
        "state_id": 2357138,
        "duration": 26,
        "state_type": "Support Ticket",
        "started": {
            "timestamp": "2015-04-08T11:50:46.287Z",
            "media_type": "mobile",
            "application_id": 5000
        },
        "completed": {
```

```
        "timestamp": "2015-04-08T11:50:46.313Z"
    }
}
```

In the above code snippet, the **CRMData** extension contains two fields: **caseNumber** and **caseOwner**:

```
"CRMData": {
    "caseNumber": "GF4GHL",
    "caseOwner": "Bogota 11:00 - 12:00"
        }
```

To display the **CRMData** extension in the state boxes of the UI, edit `profileconfig.json` and find the "STATE_EXPRESSIONS" section. This is where you can add an html expression used to display your extension information.

```
"STATE_EXPRESSIONS":{
    "CRMData": <state expression>
}
```

The expression can include standard HTML elements and angular expression mapped with the extension fields, available in the additionalInfo object.

- If your extension is not an object, you can access its value by calling `additionalInfo.values`.

- If your extension is an object, you can access field values by calling `additionalInfo.values.<field_name>`.

For instance, the fields of CRMData are available in additionalInfo.value.caseOwner and additionalInfo.value.caseNumber variables. The following code snippet shows how to display these fields in the state boxes of the UI.

```
"STATE_EXPRESSIONS":{
    "CRMData": "<p> {{additionalInfo.value.caseOwner}}'s case number is
{{additionalInfo.value.caseNumber}} </p>"
}
```


Customizing Timeline Icons for Service, Event, and Task

Configuring new Timeline Icons and Colors

1. In the **business-attributes** section of your GMS application, configure map-names = true

2. Define **media_type=MediaType** for Service, State, and Task as follows: #:
   `Service.media_type=MediaType`
   `State.media_type=MediaType`

   `Task.media_type=MediaType`

   See (resourcename.fieldname) for further details.

3. Find the **Business Attributes** in your Environment. Create the MediaType Business Attribute.

4. In MediaType, add **New Business Attribute Values** for media names such as `voice`, `ivr`, `agent`,

email, and so on. For example, create the Business Attribute Value `voice` where the name field is voice.

5. Modify and replace `TIMELINE_ICONS` and `TIMELINE_COLORS` with the MediaType names and, if required, the icons in the file located in
   ```
   <Installation folder>\gsg-app\gsg-server\gsg-web\src\main\webapp\develop\app\
   configuration\configuration.json
   ```

For example:

```
"TIMELINE_ICONS": {
 "voice" : "fonticon icon-audio-voice",
 "voip" : "fonticon icon-agent",
 "email" : "fonticon icon-email",
 "vmail" : "fonticon icon-voicemail",
 "smail" : "fonticon icon-email",
 "chat" : "fonticon icon-chat",
 "video" : "fonticon icon-chat-video",
 "cobrowsing" : "fonticon icon-cobrowse",
 "whiteboard" : "fonticon icon-checkbox",
 "appsharing" : "fonticon icon-share",
 "webform" : "fonticon icon-page-single",
 "workitem" : "fonticon icon-agent-status-work",
 "callback" : "fonticon icon-iw-active-circle-callback",
 "fax" : "fonticon icon-printer",
 "imchat" : "fonticon icon-people-chat",
 "busevent" : "fonticon icon-event-cluster",
 "alert" : "fonticon icon-alert-triangle",
 "sms" : "fonticon icon-sms-transfer",
 "any" : "fonticon icon-help",
 "auxwork" : "fonticon icon-agent-status-work",
 "outboundpreview" : "fonticon icon-outbound",
 "trainingitem" : "fonticon icon-training",
 "smssession" : "fonticon icon-sms-message",
 "mms" : "fonticon icon-inspect",
 "mmssession" : "fonticon icon-inspect-chat",
 "default" : "fonticon icon-agent-status-queue-empty"
 },

"TIMELINE_COLORS": {
 "voice" : "#203B73",
 "voip": "#2E69DB",
 "email" : "#5E99FF",
 "vmail" : "#9BBCE0",
 "smail" : "#5A6B8C",
 "chat" : "#0F6A51",
 "video" : "#569180",
 "cobrowsing" : "#14819C",
 "whiteboard" : "#7EC0C2",
 "appsharing" : "#AFD6D2",
 "webform" : "#584FB3",
 "workitem" : "#7272E0",
 "callback" : "#B9B9F0",
 "fax" : "#575746",
 "imchat" : "#827C75",
 "busevent" : "#C9C4B7",
 "alert" : "#8C6542",
 "sms" : "#8A4D67",
 "any" : "#C48C88",
 "auxwork" : "#EBC8BE",
 "outboundpreview" : "#724787",
 "trainingitem" : "#B07EC2",
```

```
 "smssession" : "#D1B4D9",
 "mms" : "#555D66",
 "mmssession" : "#4AC764",
 "default" : "#555D66"
 },
```

Providing the media_type Parameter in GMS Requests

As a result, the timeline uses the customized icons to display GMS requests that include one of
MediaType Business Attribute Values in the `media_type` parameter.

- If the `media_type` parameter includes an invalid value, the request fails.

- By default, if the request is missing the `media_type` parameter, the timeline uses the icon
  corresponding to "media_type": "any" .

- If you provide a valid `media_type` which is not mapped with any value in the `TIMELINE_ICONS` and
  `TIMELINE_COLORS` arrays, the timeline uses the icon corresponding to "media_type": "default".

List of Supported Icons

Here is the list of available icons:

```
icon-24-control-pause
icon-24-control-pause-solid
icon-24-control-play
icon-24-control-play-solid
icon-24-graph-bar
icon-24-graph-edit
icon-24-graph-grid
icon-24-graph-line
icon-24-graph-stack
icon-8-dropdown-arrow-small
icon-access
icon-actionable
icon-actionable-disable
icon-actionable-notset
icon-add
icon-add-bold
icon-adjust
icon-agent
icon-agent-add
icon-agent-edit
icon-agent-group
icon-agent-offline
icon-agent-ready
icon-agent-ready-partial
icon-agent-search
icon-agent-status-busy
icon-agent-status-not-ready
icon-agent-status-out-of-service
icon-agent-status-queue-empty
icon-agent-status-queue-full
icon-agent-status-queue-part-full
icon-agent-status-ready
icon-agent-status-ready-partial
icon-agent-status-unavailable
icon-agent-status-unknown
icon-agent-status-work
icon-agent-time
```

```
icon-agent-unavailable
icon-aix
icon-alert-checkmark
icon-alert-circle
icon-alert-info
icon-alert-octo
icon-alert-triangle
icon-app-all
icon-app-chart
icon-app-generic
icon-app-layout
icon-app-table
icon-arrow-down
icon-arrow-left
icon-arrow-right
icon-arrow-up
icon-attach
icon-audio-disable
icon-audio-music
icon-audio-no
icon-audio-rec
icon-audio-rec-disable
icon-audio-voice
icon-audio-yes
icon-bar-graph-variable-1
icon-bar-graph-variable-2
icon-biz-attribute
icon-biz-attribute-disable
icon-blocks
icon-book-address
icon-book-open-details
icon-briefcase
icon-calculator
icon-calendar-day
icon-calendar-generic
icon-calendar-month
icon-calendar-month-highlight
icon-campaign
icon-campaign-disable
icon-cells
icon-certificate
icon-chat
icon-chat-add
icon-chat-disable
icon-chat-edit
icon-chat-edit-disable
icon-chat-end
icon-chat-forward
icon-chat-multi
icon-chat-oval
icon-chat-oval-close
icon-chat-oval-multi
icon-chat-oval-transfer
icon-chat-video
icon-checkbox
icon-checkbox-tick
icon-checkmark-bold
icon-chevron-left
icon-chevron-right
icon-circle-add
icon-circle-arrow-down
icon-circle-arrow-left
icon-circle-arrow-right
```

```
icon-circle-arrow-up
icon-circle-close
icon-circle-expand-down
icon-circle-expand-left
icon-circle-expand-right
icon-circle-expand-up
icon-clip-approve
icon-clip-exchange
icon-clip-list
icon-clock
icon-clone
icon-close
icon-close-bold
icon-cloud-a
icon-cloud-b
icon-cobrowse
icon-cobrowse-chat
icon-cobrowse-end
icon-collapse
icon-column-direction
icon-column-direction-active
icon-configuration
icon-configuration-disable
icon-contact-add
icon-contact-assign
icon-contact-delete
icon-contact-id
icon-contact-id-fetch
icon-contact-switch
icon-control-backward
icon-control-forward
icon-control-left
icon-control-right
icon-control-skip-left
icon-control-skip-next
icon-control-skip-previous
icon-control-skip-right
icon-control-stop
icon-cs-mobile
icon-cs-social
icon-cs-web
icon-dash
icon-dialpad
icon-doc-add
icon-doc-all
icon-doc-all-save
icon-doc-detail
icon-doc-detail-add
icon-doc-forward
icon-doc-generic
icon-doc-generic-b
icon-doc-no
icon-doc-phone-call
icon-doc-save
icon-download
icon-dropdown-arrow
icon-email
icon-email-delete
icon-email-forward
icon-email-move-to-folder
icon-email-reply
icon-email-reply-all
icon-email-resend
```

```
icon-email-send
icon-email-transfer
icon-enterprise-extension
icon-error-bold
icon-escalate
icon-event-cluster
icon-expand
icon-expand-down
icon-expand-left
icon-expand-right
icon-expand-up
icon-face-happy
icon-face-neutral
icon-face-sad
icon-face-solid-happy
icon-face-solid-sad
icon-face-unknown
icon-facebook-square
icon-film
icon-film-broadcast
icon-filter
icon-folder
icon-folder-add
icon-folder-disabled
icon-folder-media
icon-folder-open
icon-folder-open-in
icon-folder-open-out
icon-folder-progress
icon-follow
icon-follow-disable
icon-font-bold
icon-font-decrease
icon-font-increase
icon-font-italic
icon-font-style
icon-font-underline
icon-full-screen
icon-generic
icon-generic-disable
icon-generic-laptop
icon-generic-mobile-phone
icon-generic-pc
icon-generic-tablet
icon-grab
icon-graph-bar
icon-graph-chart
icon-graph-spark
icon-grip-horizontal
icon-grip-vertical
icon-group
icon-help
icon-home
icon-host
icon-hp
icon-in
icon-in-small
icon-inspect
icon-inspect-chat
icon-inspect-chat-disable
icon-inspect-disable
icon-ivr
icon-ivr-disable
```

```
icon-iw-active-circle-callback
icon-iw-active-circle-campaign
icon-iw-active-circle-chat-round
icon-iw-active-circle-chat-square
icon-iw-active-circle-doc
icon-iw-active-circle-email
icon-iw-active-circle-facebook
icon-iw-active-circle-globe
icon-iw-active-circle-media
icon-iw-active-circle-phone-cancel
icon-iw-active-circle-phone-conf
icon-iw-active-circle-phone-dial
icon-iw-active-circle-phone-pause
icon-iw-active-circle-phone-play
icon-iw-active-circle-phone-voice
icon-iw-active-circle-question
icon-iw-active-circle-rss
icon-iw-active-circle-twitter
icon-iw-active-circle-voice
icon-iw-active-circle-voice-chat
icon-iw-active-square-view
icon-iw-arrow-outline-in
icon-iw-circle-no-add
icon-iw-circle-no-chevron-down
icon-iw-circle-no-chevron-up
icon-iw-circle-yes-add
icon-iw-circle-yes-chevron-down
icon-iw-circle-yes-chevron-up
icon-iw-control-circle-back
icon-iw-control-circle-fwd
icon-iw-control-circle-home
icon-iw-control-circle-play
icon-iw-control-circle-stop
icon-iw-done-and-stop
icon-iw-history
icon-iw-in-progress
icon-iw-inspect-voice
icon-iw-inspect-voice-cancel
icon-iw-monitor-chat
icon-iw-monitor-voice
icon-iw-queue
icon-iw-routing-point
icon-iw-save
icon-iw-save-as
icon-iw-square-facebook-email
icon-iw-square-facebook-no
icon-iw-square-facebook-transfer
icon-iw-square-facebook-workbin
icon-iw-square-rss-transfer
icon-iw-square-rss-workbin
icon-iw-square-twitter-follow
icon-iw-square-twitter-no
icon-iw-square-twitter-transfer
icon-iw-square-twitter-workbin
icon-iw-video-suspend
icon-iw-video-thumb-cancel
icon-iw-video-thumb-show
icon-iw-video-thumb-swap
icon-iw-voice-inspect
icon-iw-voice-inspect-cancel
icon-link
icon-linux
icon-list-bullets
```

```
icon-list-numbers
icon-login-checkmark
icon-login-error
icon-login-info
icon-merge
icon-minimize
icon-more
icon-ms
icon-network
icon-notebook
icon-out
icon-out-small
icon-outbound
icon-page-multi
icon-page-search
icon-page-single
icon-palette
icon-parameter
icon-parameter-group
icon-pencil
icon-people
icon-people-chat
icon-person
icon-phone
icon-phone-add
icon-phone-chat-media
icon-phone-conference
icon-phone-decline
icon-phone-disabled
icon-phone-forward
icon-phone-hold
icon-phone-incoming
icon-phone-pause
icon-phone-rec
icon-phone-rec-pause
icon-phone-rec-resume
icon-phone-rec-stop
icon-phone-voice
icon-phone-voice-retrieve
icon-phones
icon-picture
icon-pie-chart
icon-pin
icon-pin-alert
icon-pin-check
icon-pin-sold-remove
icon-pin-solid
icon-pin-solid-add
icon-pin-time
icon-pin-zoom-in
icon-pin-zoom-out
icon-pinned
icon-play-list
icon-printer
icon-progress-bar
icon-radio
icon-radio-active
icon-refresh
icon-refresh-with-feedback
icon-reload
icon-remember
icon-reset
icon-rss
```

```
icon-rss-square
icon-screen-rec-pause
icon-screen-rec-record
icon-screen-rec-resume
icon-screen-rec-stop
icon-search
icon-search-next
icon-search-previous
icon-searched-term
icon-secure
icon-select
icon-select-no
icon-select-yes
icon-server
icon-server-add
icon-settings-gear
icon-settings-gears
icon-settings-sliders
icon-share
icon-sharethis
icon-shuffle
icon-site
icon-site-disable
icon-skills
icon-slide-back
icon-slide-fwd
icon-small-checkmark
icon-small-close
icon-small-plus
icon-sms-cancel
icon-sms-message
icon-sms-reply
icon-sms-transfer
icon-solaris
icon-special-g-brandmark
icon-star-edit
icon-star-outline
icon-stars
icon-stars-add
icon-stars-lock
icon-stars-remove
icon-superviser-chat-cancel
icon-supervisor
icon-supervisor-cancel
icon-supervisor-chat
icon-supervisor-voice
icon-supervisor-voice-cancel
icon-switch
icon-switch-disable
icon-symbol-man
icon-symbol-woman
icon-tag-stat-add
icon-tag-stat-remove
icon-team-communicator
icon-tenant
icon-tenant-disable
icon-text-centered
icon-text-generic
icon-text-indent-left
icon-text-indent-right
icon-text-justify-center
icon-text-justify-full
icon-text-justify-left
```

```
icon-text-justify-right
icon-text-spelling
icon-thumbs-down
icon-thumbs-up
icon-tick
icon-timeline
icon-toggle
icon-toggle-off
icon-toggle-on
icon-training
icon-trash
icon-twitter
icon-twitter-square
icon-unmerge
icon-video
icon-video-add
icon-video-disable
icon-video-end
icon-video-forward
icon-video-pause
icon-videocam-1
icon-videocam-1-disable
icon-videocam-2
icon-videocam-2-disable
icon-view-details-bottom
icon-view-details-right
icon-view-grid
icon-view-list
icon-view-stacked
icon-view-tree
icon-window-expand
icon-zoom-in
icon-zoom-out
```

Customize Journey Dashboard for Pulse

Pulse widgets for Context Services are not editable from the Pulse Template Management UI. To customize widgets for Context Services, you must edit and submit the templates through the Pulse REST API.

> ### Important
> Starting in 8.5.228.02, due to JSON library updates, the boolean and integer values are processed as strings and should be provided within double quotes ("").

Prerequisites

Component	Version
GMS	8.5.106+
Pulse	8.5.102.xx or 8.5.105.01 **Important** Versions 8.5.103 and 8.5.104 are not supported.
GAX	8.5.102.18

Set Pulse Permissions
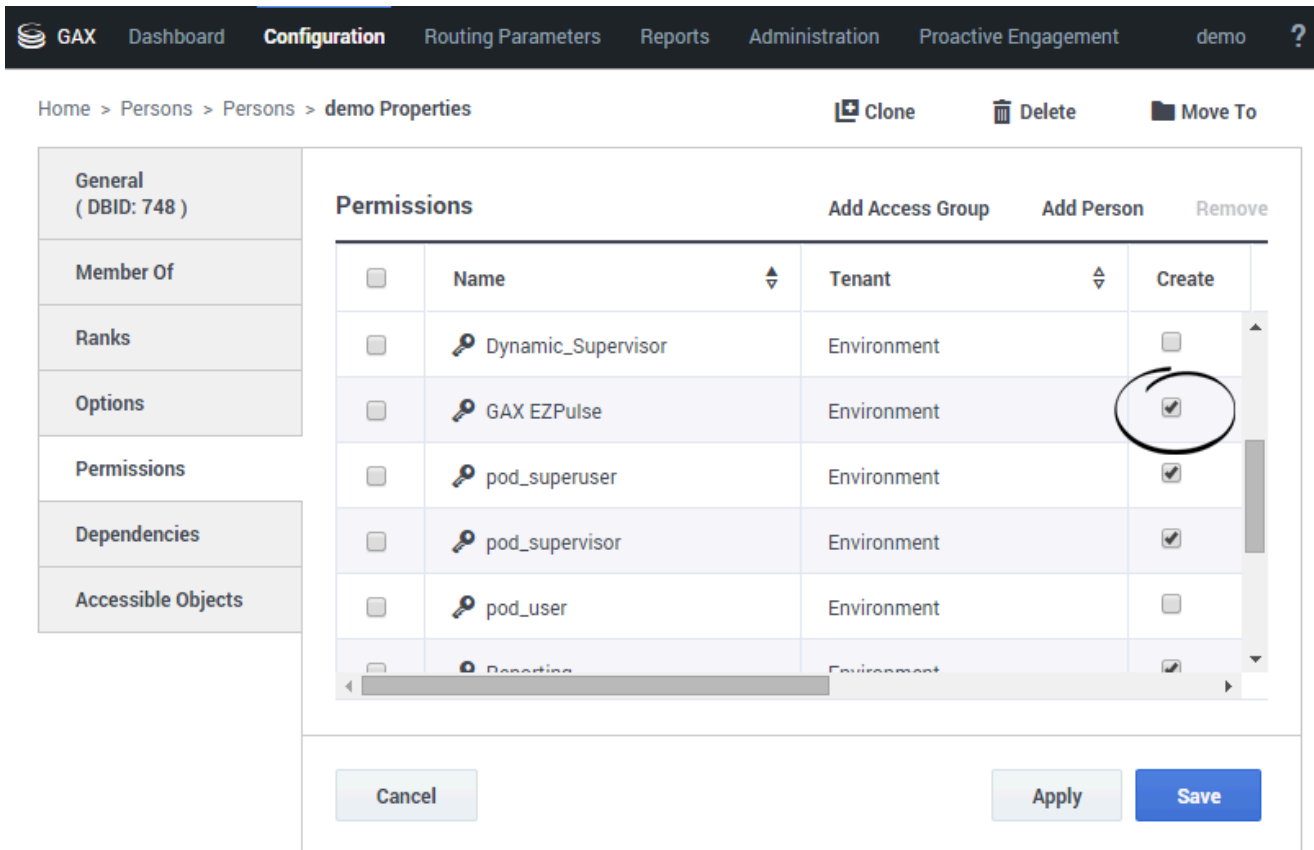
Open GAX or Genesys Administrator and navigate to **Configuration > Persons**. Select your user and edit his or her permissions to add the **GAX EZPulse** permissions, then **Save**.

Create Journey Pulse Widgets

Add journey sample layouts to Pulse

Context Services Pulse layout samples are available in the following directory:

- `<GMS installation directory>/`**Pulse_8.5.102** for integration with Pulse 8.5.102 versions.
- `<GMS installation directory>/`**Pulse_8.5.103** for integration with Pulse 8.5.105.01.

To deploy these samples in Pulse, you can use the Genesys Pulse Restful Web Service API.
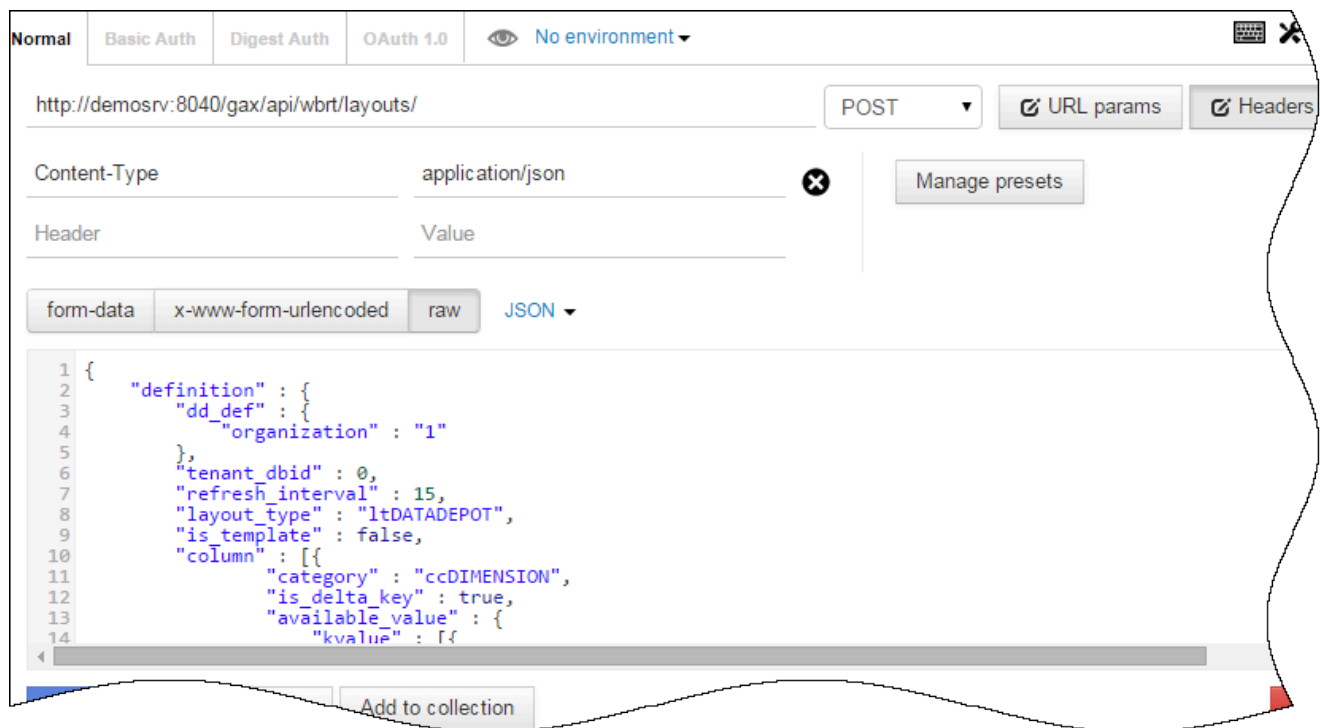
Start a Restful client application (Postman in our example). Submit a POST or PUT API query to GAX to publish the layout sample.

- For Pulse 8.5.102 versions:
    - POST /api/wbrt/layouts/ to deploy a new layout.
    - PUT /api/wbrt/layouts/<id> to update an existing layout.
- For Pulse 8.5.105.01:
    - POST /api/wbrt/templates/ to deploy a new layout.
    - PUT /api/wbrt/templates/<guid> to update an existing layout.

Add your widgets

For Pulse 8.5.102 versions
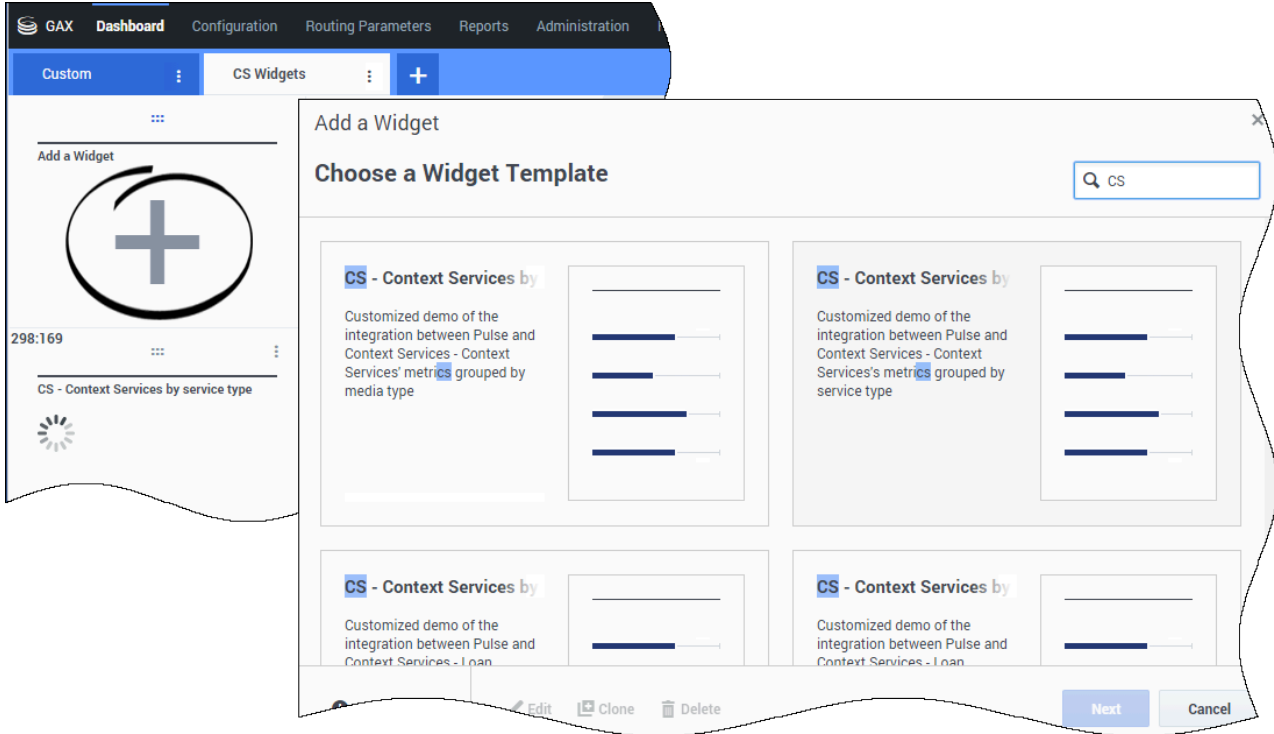
Once the templates are published in Pulse, it's very easy. Just navigate to your Pulse Dashboard. The new layouts will be available in the creation wizard.

For Pulse 8.5.105.01 version
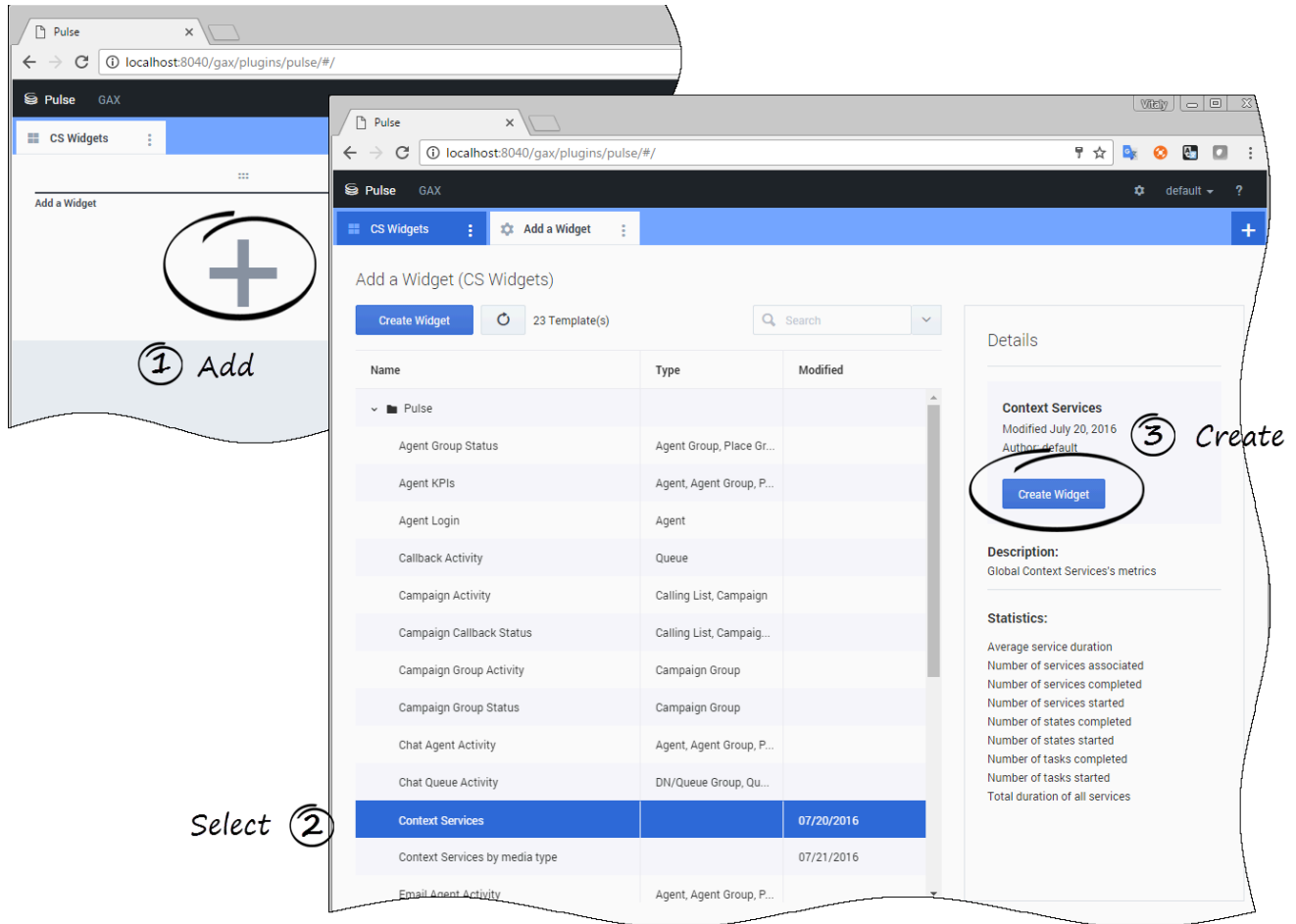
Once the templates are published in Pulse, it's very easy. Just navigate to your Pulse Dashboard. The new layouts will be available in the creation wizard.

Edit Pulse External Templates

> ## Important
> Starting in 8.5.228.02, due to JSON library updates, the boolean and integer values are processed as strings and should be provided within double quotes ("").

Retrieve your layout for Pulse 8.5.102
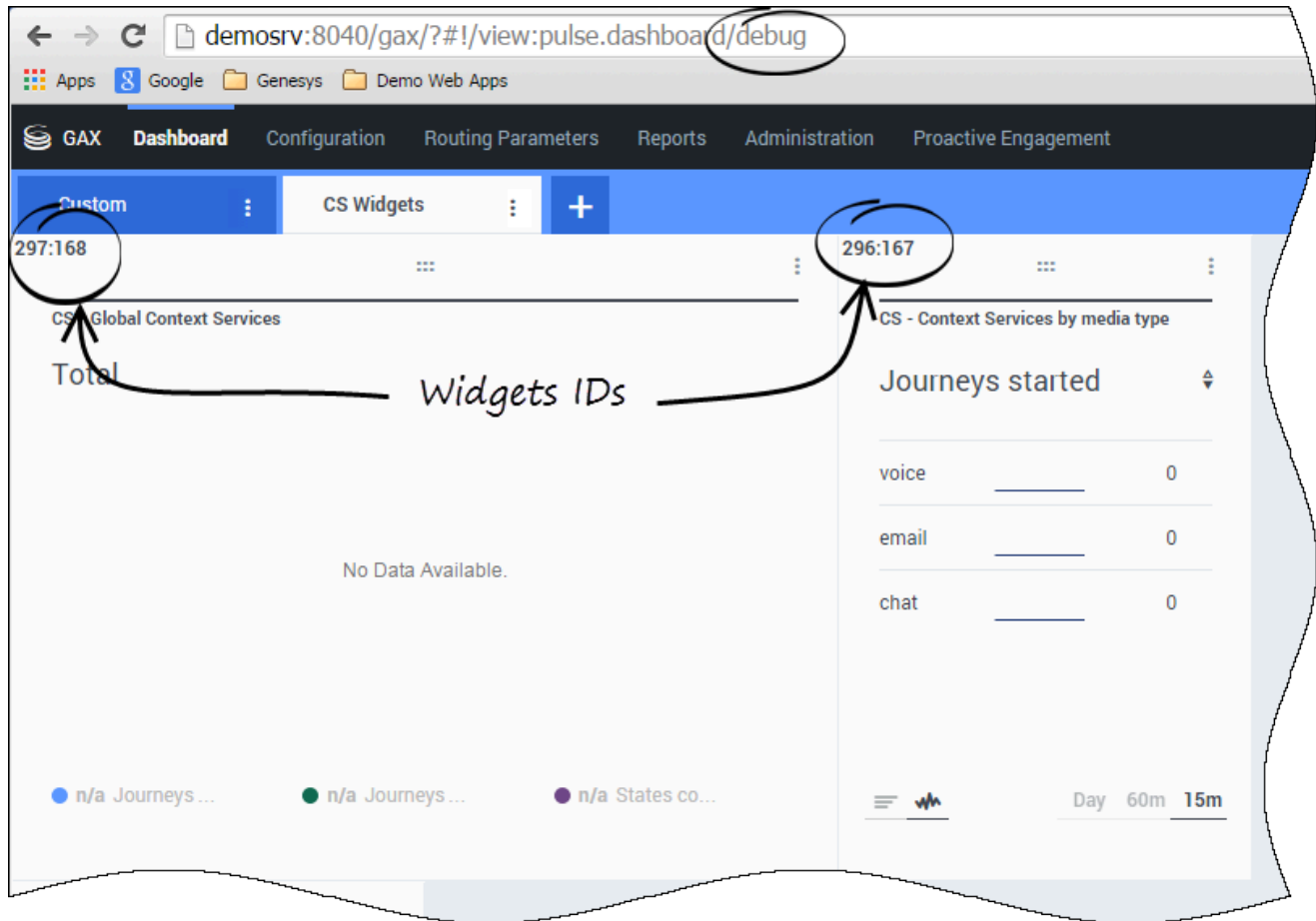
Open Pulse in Genesys Administrator extensions and navigate to your Context Services widget. Append /debug at the end of your URL. The widget and layout IDs will show up in the left corner of each widget.

In our example, 297:168 means that the widget ID is 297 and the layout ID is 168.

Get the layout definition of your widget
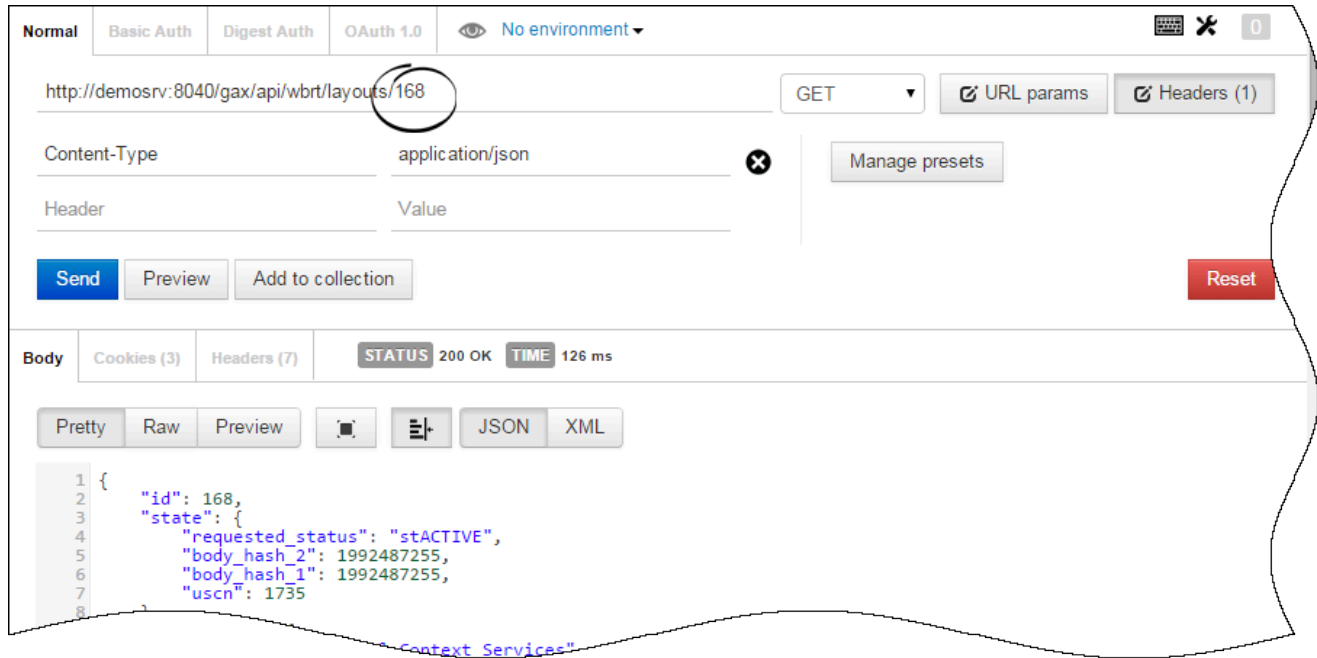
Start a Restful client application (Postman in our example). Submit a GET API query to GAX to retrieve the layout associated to your widget ID:
GET `http://<host>:<port>/gax/api/wbrt/layouts/<layoutID>` URL with the header set to
`Content-Type= application/json` See the Genesys Pulse Restful Web Service API for further details.

[+] Expand the layout example

```
{
    "id": 169,
    "state": {
        "requested_status": "stACTIVE",
        "body_hash_2": -383534282,
        "body_hash_1": -383534282,
        "uscn": 1737
    },
    "definition": {
        "name": "CS - Context Services by service type",
        "description": "Customized demo
of the integration between Pulse and Context Services
- Context Services's metrics grouped by service type",
        "refresh_interval": 15,
        "tenant_dbid": 1,
        "layout_type": "ltDATADEPOT",
        "enable_delta_snapshots": false,
        "default_widget": {
            "id": 0,
            "size": "1x2",
            "label": "CS - Context Services by service type",
            "view": [
                {
                    "type": "BarView",
```

```
                    "column_selector": [
                        "service_started"
                    ],
                    "sorting": [
                        {
                            "is_asc": false
                        }
                    ]
                }
            ]
        },
        "template_layout_id": 25,
        "template_column": [
            {
                "id": "service_started",
                "type": "ctDATADEPOT",
                "format": "integer",
                "category": "ccMEASURE",
                "label": "started",
                "vt": "vINT",
                "dd_column": {
                    "group_name": "cs.service",
                    "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                    "aggregate_name": "service.started",
                    "rollup_range": "DAY",
                    "calculation_type": "COUNT",
                    "criteria_name": "service_type"
                }
            },
            {
                "id": "service_started_perc",
                "type": "ctDATADEPOT",
                "format": "percent",
                "category": "ccMEASURE",
                "label": "% started",
                "vt": "vDBL",
                "dd_column": {
                    "group_name": "cs.service",
                    "expr": "#a = #dd_stat(#dd, #organization, #group_name,
#aggregate_name, {#criteria_name, #selected}, #rollup_range, #calculation_type);
#b = #dd_stat(#dd, #organization, #group_name, #aggregate_name, {},
#rollup_range, #calculation_type); #a = T(java.lang.Double).parseDouble(#a); #b =
T(java.lang.Double).parseDouble(#b); (#b == 0) ? 0 : #a*100/#b",
                    "aggregate_name": "service.started",
                    "rollup_range": "DAY",
                    "calculation_type": "COUNT",
                    "criteria_name": "service_type"
                }
            },
            {
                "id": "service_completed",
                "type": "ctDATADEPOT",
                "format": "integer",
                "category": "ccMEASURE",
                "label": "completed",
                "vt": "vINT",
                "dd_column": {
                    "group_name": "cs.service",
                    "expr": "#a = #dd_stat(#dd, #organization, #group_name,
#aggregate_name, {#criteria_name, #selected}, #rollup_range, #calculation_type)",
                    "aggregate_name": "service.completed",
                    "rollup_range": "DAY",
```

```
                "calculation_type": "COUNT",
                "criteria_name": "service_type"
            }
        },
        {
            "id": "service_completed_perc",
            "type": "ctDATADEPOT",
            "format": "percent",
            "category": "ccMEASURE",
            "label": "% completed",
            "vt": "vDBL",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type); #b = #dd_stat(#dd,
#organization, #group_name, #aggregate_name, {}, #rollup_range, #calculation_type);
#a = T(java.lang.Double).parseDouble(#a); #b = T(java.lang.Double).parseDouble(#b); (#b == 0)
? 0 : #a*100/#b",
                "aggregate_name": "service.completed",
                "rollup_range": "DAY",
                "calculation_type": "COUNT",
                "criteria_name": "service_type"
            }
        },
        {
            "id": "service_associated",
            "type": "ctDATADEPOT",
            "format": "integer",
            "category": "ccMEASURE",
            "label": "associated",
            "vt": "vINT",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                "aggregate_name": "service.associated",
                "rollup_range": "DAY",
                "calculation_type": "COUNT",
                "criteria_name": "service_type"
            }
        },
        {
            "id": "service_durationAVG",
            "type": "ctDATADEPOT",
            "format": "time",
            "category": "ccMEASURE",
            "label": "Avg Duration",
            "vt": "vINT",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                "aggregate_name": "service.durationAVG",
                "rollup_range": "DAY",
                "calculation_type": "AVG",
                "criteria_name": "service_type"
            }
        },
        {
            "id": "service_durationSUM",
            "type": "ctDATADEPOT",
            "format": "time",
            "category": "ccMEASURE",
```

```
                "label": "Duration",
                "vt": "vINT",
                "dd_column": {
                    "group_name": "cs.service",
                    "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                    "aggregate_name": "service.durationSUM",
                    "rollup_range": "DAY",
                    "calculation_type": "SUM",
                    "criteria_name": "service_type"
                }
            },
            {
                "id": "month_service_started",
                "type": "ctDATADEPOT",
                "format": "integer",
                "category": "ccMEASURE",
                "label": "started (30 days)",
                "vt": "vINT",
                "dd_column": {
                    "group_name": "cs.service",
                    "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                    "aggregate_name": "service.started",
                    "rollup_range": "MONTH",
                    "calculation_type": "COUNT",
                    "criteria_name": "service_type"
                }
            },
            {
                "id": "month_service_completed",
                "type": "ctDATADEPOT",
                "format": "integer",
                "category": "ccMEASURE",
                "label": "completed (30 days)",
                "vt": "vINT",
                "dd_column": {
                    "group_name": "cs.service",
                    "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                    "aggregate_name": "service.completed",
                    "rollup_range": "MONTH",
                    "calculation_type": "COUNT",
                    "criteria_name": "service_type"
                }
            },
            {
                "id": "month_service_completed_perc",
                "type": "ctDATADEPOT",
                "format": "percent",
                "category": "ccMEASURE",
                "label": "% completed (30 days)",
                "vt": "vDBL",
                "dd_column": {
                    "group_name": "cs.service",
                    "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type); #b = #dd_stat(#dd,
#organization, #group_name, #aggregate_name, {}, #rollup_range, #calculation_type); #a =
T(java.lang.Double).parseDouble(#a); #b = T(java.lang.Double).parseDouble(#b); (#b == 0) ? 0
: #a*100/#b",
                    "aggregate_name": "service.completed",
                    "rollup_range": "MONTH",
                    "calculation_type": "COUNT",
```

```
                    "criteria_name": "service_type"
                }
            }
        ],
        "column": [
            {
                "id": "_Object$ID",
                "category": "ccDIMENSION",
                "selected_value": {
                    "type": "stKEYVAL",
                    "kvalue": [
                        {
                            "k": "4480",
                            "v": "Auto Care Application"
                        },
                        {
                            "k": "4487",
                            "v": "Auto Care Cancelation"
                        },
                        {
                            "k": "4481",
                            "v": "Auto Care Selection"
                        },
                        {
                            "k": "4457",
                            "v": "Branch Appointment"
                        },
                        {
                            "k": "4479",
                            "v": "Credit Card Application"
                        },
                        {
                            "k": "4485",
                            "v": "Credit Card Cancelation"
                        },
                        {
                            "k": "4478",
                            "v": "Credit Card Selection"
                        },
                        {
                            "k": "4458",
                            "v": "General Inquiries"
                        },
                        {
                            "k": "4486",
                            "v": "Home Care Cancelation"
                        },
                        {
                            "k": "4482",
                            "v": "Home Care Selection"
                        },
                        {
                            "k": "4459",
                            "v": "Loan Application"
                        },
                        {
                            "k": "4484",
                            "v": "Loan Cancelation"
                        },
                        {
                            "k": "4460",
                            "v": "Loan Selection"
                        }
```

```
                    ]
                },
                "is_delta_key": true,
                "available_value": {
                    "type": "stKEYVAL",
                    "kvalue": [
                        {
                            "k": "4460",
                            "v": "Loan Selection"
                        },
                        {
                            "k": "4457",
                            "v": "Branch Appointment"
                        },
                        {
                            "k": "4478",
                            "v": "Credit Card Selection"
                        },
                        {
                            "k": "4479",
                            "v": "Credit Card Application"
                        },
                        {
                            "k": "4459",
                            "v": "Loan Application"
                        },
                        {
                            "k": "4485",
                            "v": "Credit Card Cancelation"
                        },
                        {
                            "k": "4486",
                            "v": "Home Care Cancelation"
                        },
                        {
                            "k": "4487",
                            "v": "Auto Care Cancelation"
                        },
                        {
                            "k": "4481",
                            "v": "Auto Care Selection"
                        },
                        {
                            "k": "4480",
                            "v": "Auto Care Application"
                        },
                        {
                            "k": "4482",
                            "v": "Home Care Selection"
                        },
                        {
                            "k": "4484",
                            "v": "Loan Cancelation"
                        },
                        {
                            "k": "4458",
                            "v": "General Inquiries"
                        }
                    ]
                }
            },
            {
                "id": "_Object$Name",
```

```
                    "format": "string",
                    "category": "ccDIMENSION"
                },
                {
                    "id": "service_started",
                    "type": "ctDATADEPOT",
                    "format": "integer",
                    "category": "ccMEASURE",
                    "label": "started",
                    "vt": "vINT",
                    "dd_column": {
                        "group_name": "cs.service",
                        "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                        "aggregate_name": "service.started",
                        "rollup_range": "DAY",
                        "calculation_type": "COUNT",
                        "criteria_name": "service_type"
                    }
                },
                {
                    "id": "service_started_perc",
                    "type": "ctDATADEPOT",
                    "format": "percent",
                    "category": "ccMEASURE",
                    "label": "% started",
                    "vt": "vDBL",
                    "dd_column": {
                        "group_name": "cs.service",
                        "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type); #b = #dd_stat(#dd,
#organization, #group_name, #aggregate_name, {}, #rollup_range, #calculation_type); #a =
T(java.lang.Double).parseDouble(#a); #b = T(java.lang.Double).parseDouble(#b); (#b == 0) ? 0
: #a*100/#b",
                        "aggregate_name": "service.started",
                        "rollup_range": "DAY",
                        "calculation_type": "COUNT",
                        "criteria_name": "service_type"
                    }
                },
                {
                    "id": "service_completed",
                    "type": "ctDATADEPOT",
                    "format": "integer",
                    "category": "ccMEASURE",
                    "label": "completed",
                    "vt": "vINT",
                    "dd_column": {
                        "group_name": "cs.service",
                        "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                        "aggregate_name": "service.completed",
                        "rollup_range": "DAY",
                        "calculation_type": "COUNT",
                        "criteria_name": "service_type"
                    }
                },
                {
                    "id": "service_completed_perc",
                    "type": "ctDATADEPOT",
                    "format": "percent",
                    "category": "ccMEASURE",
                    "label": "% completed",
```

```
            "vt": "vDBL",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type); #b = #dd_stat(#dd,
#organization, #group_name, #aggregate_name, {}, #rollup_range, #calculation_type); #a =
T(java.lang.Double).parseDouble(#a); #b = T(java.lang.Double).parseDouble(#b); (#b == 0) ? 0
: #a*100/#b",
                "aggregate_name": "service.completed",
                "rollup_range": "DAY",
                "calculation_type": "COUNT",
                "criteria_name": "service_type"
            }
        },
        {
            "id": "service_associated",
            "type": "ctDATADEPOT",
            "format": "integer",
            "category": "ccMEASURE",
            "label": "associated",
            "vt": "vINT",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                "aggregate_name": "service.associated",
                "rollup_range": "DAY",
                "calculation_type": "COUNT",
                "criteria_name": "service_type"
            }
        },
        {
            "id": "service_durationAVG",
            "type": "ctDATADEPOT",
            "format": "time",
            "category": "ccMEASURE",
            "label": "Avg Duration",
            "vt": "vINT",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                "aggregate_name": "service.durationAVG",
                "rollup_range": "DAY",
                "calculation_type": "AVG",
                "criteria_name": "service_type"
            }
        },
        {
            "id": "service_durationSUM",
            "type": "ctDATADEPOT",
            "format": "time",
            "category": "ccMEASURE",
            "label": "Duration",
            "vt": "vINT",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                "aggregate_name": "service.durationSUM",
                "rollup_range": "DAY",
                "calculation_type": "SUM",
                "criteria_name": "service_type"
```

```
            }
        },
        {
            "id": "month_service_started",
            "type": "ctDATADEPOT",
            "format": "integer",
            "category": "ccMEASURE",
            "label": "started (30 days)",
            "vt": "vINT",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                "aggregate_name": "service.started",
                "rollup_range": "MONTH",
                "calculation_type": "COUNT",
                "criteria_name": "service_type"
            }
        },
        {
            "id": "month_service_completed",
            "type": "ctDATADEPOT",
            "format": "integer",
            "category": "ccMEASURE",
            "label": "completed (30 days)",
            "vt": "vINT",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                "aggregate_name": "service.completed",
                "rollup_range": "MONTH",
                "calculation_type": "COUNT",
                "criteria_name": "service_type"
            }
        },
        {
            "id": "month_service_completed_perc",
            "type": "ctDATADEPOT",
            "format": "percent",
            "category": "ccMEASURE",
            "label": "% completed (30 days)",
            "vt": "vDBL",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type); #b = #dd_stat(#dd,
#organization, #group_name, #aggregate_name, {}, #rollup_range, #calculation_type); #a =
T(java.lang.Double).parseDouble(#a); #b = T(java.lang.Double).parseDouble(#b); (#b == 0) ? 0
: #a*100/#b",
                "aggregate_name": "service.completed",
                "rollup_range": "MONTH",
                "calculation_type": "COUNT",
                "criteria_name": "service_type"
            }
        }
    ],
    "dd_def": {
        "organization": "1"
    }
},
"generator_info": {
    "name": "Pulse",
```
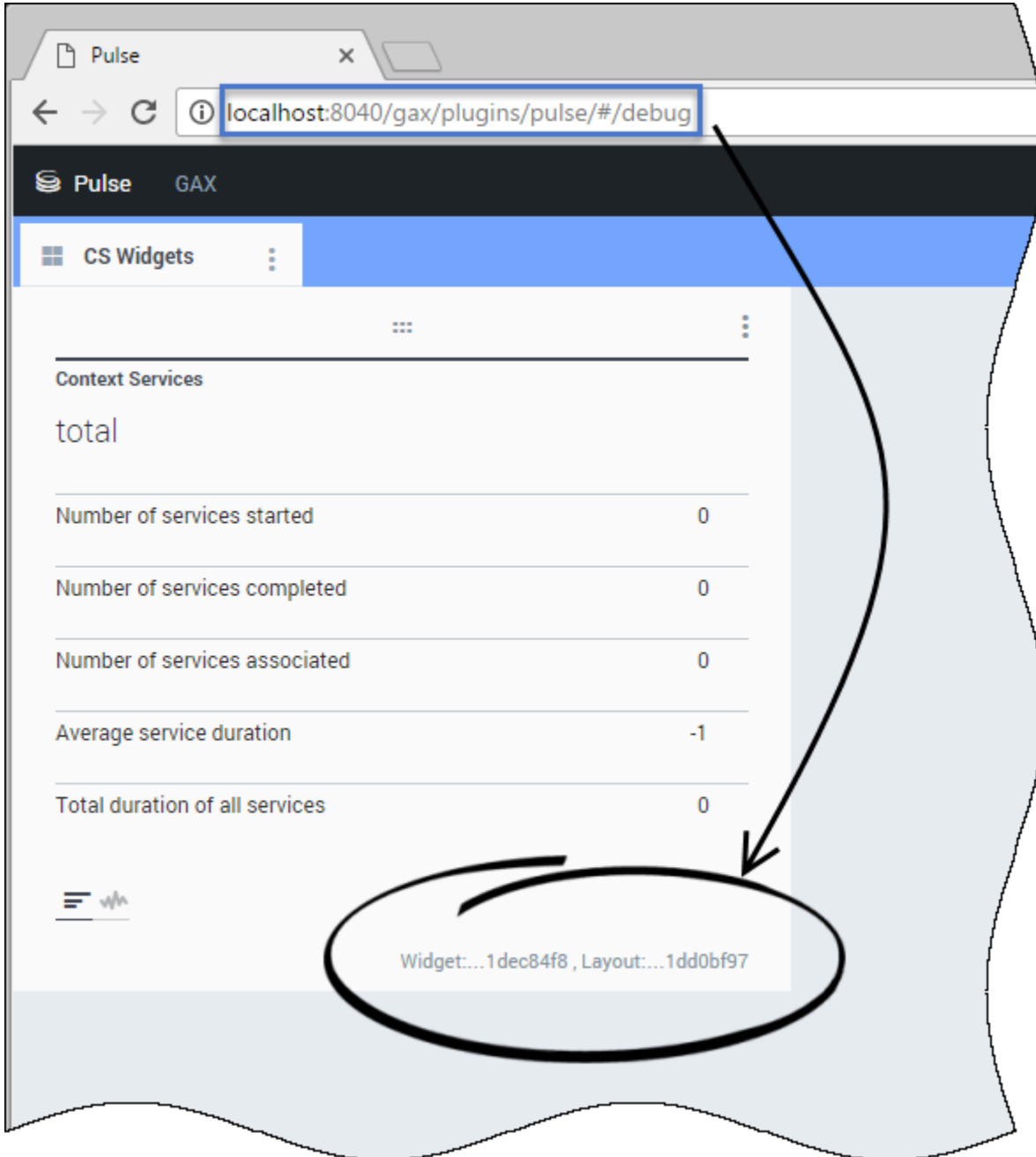
```
        "version": "8.5.101.03",
        "timestamp": 1467298003720,
        "pulse_app_id": 1,
        "cme_app_dbid": 366,
        "cme_app_name": "GAX_Server"
    }
}
```

Retrieve your layout for Pulse 8.5.105.01

Open Pulse in Genesys Administrator extensions and navigate to your Context Services widget. Append /debug at the end of your URL and press **Enter**.

- The URL is restored to its previous state (without /debug).

- The widget and layout GUIDs will show up as a clickable link available in the right bottom corner of each widget.

Click the link to display the layout in a separate tab.

Edit and publish the layout

To learn how you can modify the layout, read the detailed section below. Then, deploy or update your layout.

How to modify the JSON Layout

Each Pulse template has a unique ID, a definition, and a state property. If you wish to modify the template, do not modify the template ID; else, if you modify this ID, you will create a new template.

```
{
    "id": <id>,
    "state" : { /** specific state information - do not modify **/
    "definition": {
        /*... See Definition section */
    },
    "generator_info": { /** specific generated information - do not modify **/
    }
}
```

> ### Important
> In Pulse version 8.5.105.01, the templates use the guid field instead of id.

Definition property

In the Definition parameter of the JSON layout, you will find the following parameters.

> **Important**
>
> Not all Pulse parameters are listed below.

Field	Description
name	Defines the template name displayed to the User in the **Add a Widget** interface of Pulse.
description	Defines the template description displayed to the User in the **Add a Widget** interface of Pulse. "Customized demo of the integration between Pulse and Context Services - Context Services's metrics grouped by service type"
layout_type	Defines the source of the layout type. Do not modify this type; it must be set to `ltDATADEPOT`. DataDepot is an internal collector that Context Services uses to communicate with Pulse.
refresh_interval	Defines the refresh interval of the template in seconds. For instance, `60`.
columns	Defines the content of the objects and statistics available in the template. See the `columns` property table below for further details.

Columns property

The following table details the JSON layout columns properties of the definition attribute.
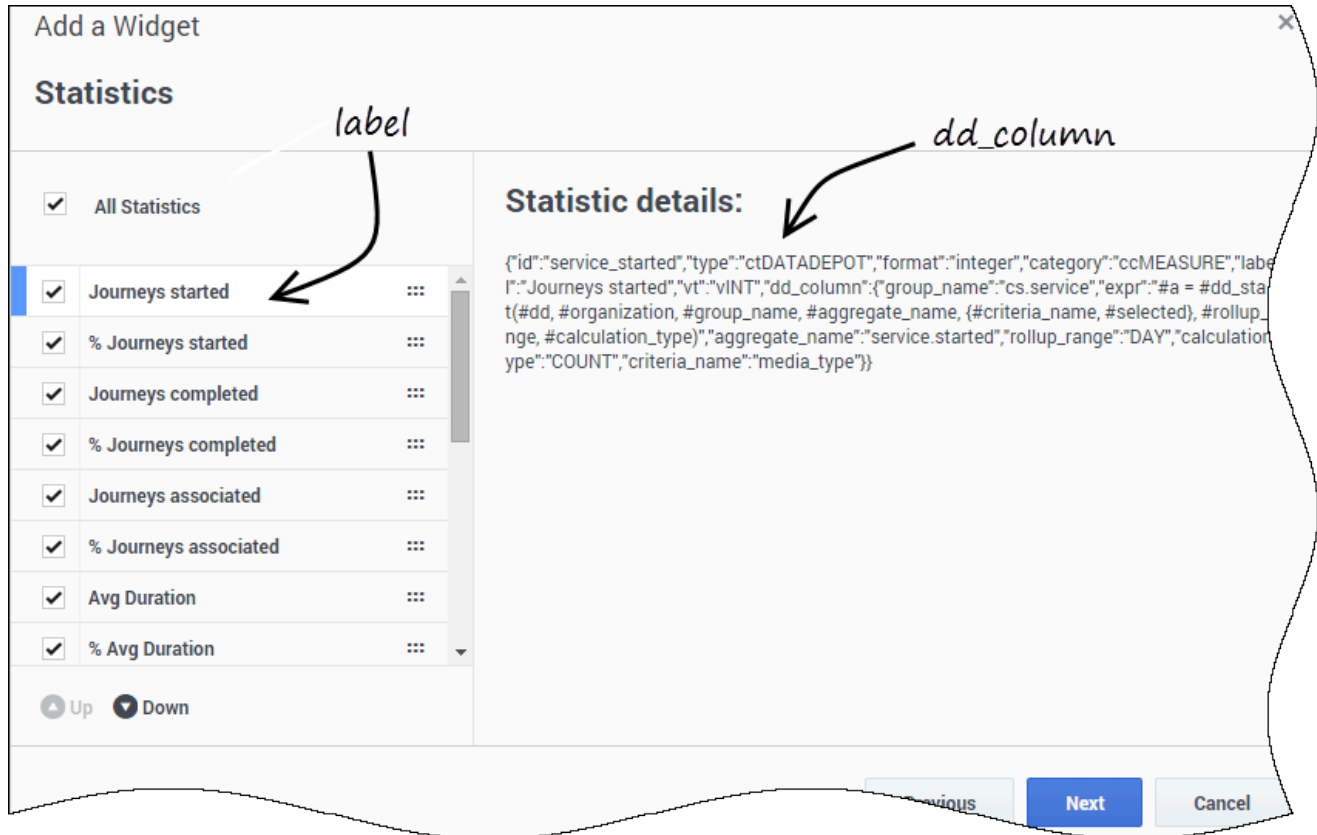
columns

Field	Description
id	Defines the technical alias of the objects or statistics IDs defined in the `template_columns` property. For instance, "id": "_Object$ID" or "id": "service_started"
category	Category of the column that can be a dimension if set to `ccDIMENSION`, or a statistic if set to `ccMEASURE`.
format	Format of this column that can be one of the following values: string, percent, integer, time
available_value	Objects available in the template. • This field is mandatory for the **Edit** wizard of the widget when the user selects the object. • The kvalue array contains key-value pairs of Business Attribute values to use in this category.

Field	Description

	• The k field contains the dbid of the business attributes.
	
	• The v field contains the label or display name to display for this Business attribute.
	
	```"available_value": {```
	``` "type": "stKEYVAL",```
	``` "kvalue": [```
	``` { "v": "Loan Selection",```
	``` "k": "4460"},```
	``` { "v": "Credit Card Selection",```
	``` "k": "4478"}```
	```]```
	``` }```

selected_value	Objects selected in the template.
	
	• This field is mandatory for the **Edit** wizard of the widget when the user selects the object.
	
	• The kvalue array contains key-value pairs of Business Attribute values to use in this category.
	
	    • The k field contains the dbid of the business attributes.
	
	    • The v field contains the label or display name to display for this Business attribute.
	
	```"selected_value": {```
	``` "type": "stKEYVAL",```
	``` "kvalue": [```
	``` { "v": "Loan Selection",```
	``` "k": "4460"},```
	``` { "v": "Credit Card Selection",```
	``` "k": "4478"}```
	```]```
	``` }```

Template_column property

For Pulse 8.5.102 versions

The `template_column` property contains an array of column templates. Each column template defines a statistic to calculate and will be displayed in the **Statistics** panel of the widget creation wizard.

```
"template_column": [
        {
            "id": "service_started",
            "type": "ctDATADEPOT",
            "format": "integer",
            "category": "ccMEASURE",
            "label": "started",
            "vt": "vINT",
            "dd_column": {
                "group_name": "cs.service",
                "expr": "#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name,
{#criteria_name, #selected}, #rollup_range, #calculation_type)",
                "aggregate_name": "service.started",
                "rollup_range": "DAY",
                "calculation_type": "COUNT",
                "criteria_name": "service_type"
            }
        },
```

```
      { /** Other statistics **/
            }
   ],
```

For Pulse 8.5.105.01 version
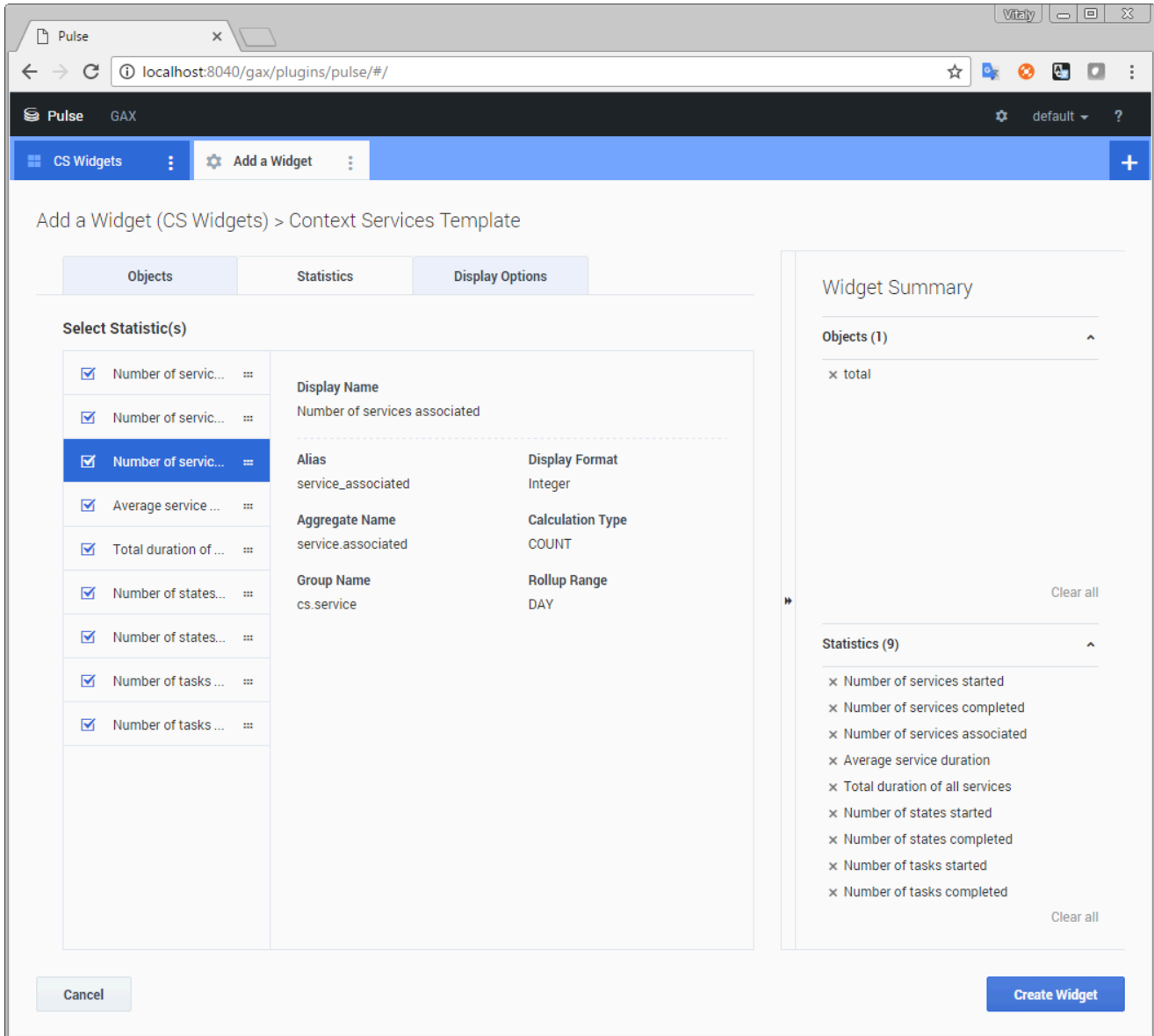
The `template_column` property contains an array of column templates. For each `dd_column`, Pulse displays details on how this column is calculated: type of calculation, roll-up period, aggregate, group names, and more.

Create dd_column objects

Each `dd_column` object is a JSON Object that defines the statistics data and calculations. The

statistics is calculated within the **expr** mandatory property that is a Spring expression. See the official documentation here.

This expression calls a custom extension function dd_stat to retrieve Statistics and assigns the result to a variable named "a". The expression uses the other properties defined in the dd_column structure to calculate the statistics.

Property Name	Possible Values	Definition
expr	Spring Expression Language (SpEL).	Defines a statistic using the above values to create a Spring expression. The expression uses the properties defined in the dd_column structure: • group_name • aggregate_name • criteria_name • calculation_type
group_name	cs.service	Group to use for Context Services.
aggregate_name	• service.started • service.completed • service.associated • state.started • state.completed • task.started • task.completed • service.duration<SUM,AVG,MIN,MAX>	Events used for the statistics.
criteria_name	• media_type • service_type • state_type • task_type	Not mandatory.
calculation_type	COUNT	Counts events.
	SUM MIN MAX AVG	Used for duration statistics (service.duration<SUM,AVG,MIN,MAX>).
rollup_range	• "QUARTER_HOUR"	Timeframe for the statistic.

Property Name	Possible Values	Definition
	• "HALF_HOUR" • "HOUR" • "DAY" • "WEEK" • "MONTH" • "QUARTER_YEAR" • "HALF_YEAR" • "YEAR"	

The dd_column example above retrieves a simple value, but it could report a percentage value. For example, you can report a percentage of services started for one media type compared to all of the started services or media types.

```
#a = #dd_stat(#dd, #organization, #group_name, #aggregate_name, {#criteria_name, #selected},
#rollup_range, #calculation_type);
#b = #dd_stat(#dd, #organization, #group_name, #aggregate_name, {}, #rollup_range,
#calculation_type);
#a = T(java.lang.Double).parseDouble(#a);
#b = T(java.lang.Double).parseDouble(#b);
(#b == 0) ? 0 : #a*100/#b
```

The code above queries two Data Depot metrics, converts each value from String to double , then calculates the required percentage value. As you may have noticed, both expressions used the #selected variable.

Metrics across multiple groups

The Pulse layout definition allows you to query for the same metrics across multiple groups (for example, media types). To do so, you can define a list of all possible values in the structure called available_value. During the widget configuration, the Pulse user will select the available values to display on the dashboard.

The values that the user selects are saved as a subset of the available values in the selected_value structure and later iterated to provide the result.

```
"available_value" : {
  "kvalue" : [{
    "v" : "media type 1",
    "k" : "1"
  }, {
    "v" : "media type 2",
    "k" : "2"
  }, {
    "v" : "media type 3",
    "k" : "3"
  }],
  "type" : "stKEYVAL"
},
"selected_value" : {
```

```
  "kvalue" : [{
    "v" : "media type 1",
    "k" : "1"
  }, {
    "v" : "media type 2",
    "k" : "2"
  }],
 "type" : "stKEYVAL"
}
```