Logo

Description automatically generatedShape, circle

Description automatically generated

**2024**

**Agora Tc Integration with External Applications using  
  
1) Real-time Mode  
2) Publish/Subscribe Mode**

TDSI - Technical Design and Solution Implementation Document

Document Details

|  |  |
| --- | --- |
| **Solution Design Document** | |
| **Project Owner** | David Cordava |
| **Document Title** | TDSI\_TcIntegration\_ExternalApplications\_Technical Design & Solution Implementation Document V1.3 |
| **Document Author** | Scott London, Sanjeev Beemidi |
| **Document Version** | 1.0 |
| **Document Status** | Issued for Comments |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Edit History** | | | | |
| **Version** | **Description** | **Name** | **Role** | **Date** |
| V1.0 | Initial draft | Scott London, Sanjeev Beemidi | Solution Architect | 08/28/2024 |

Table of Contents

[1. Introduction 4](#__RefHeading___Toc23771_1576096024)

[1.1. Purpose 4](#__RefHeading___Toc23773_1576096024)

[1.2. Scope 4](#__RefHeading___Toc23775_1576096024)

[1.3. Technical Design & Solution Implementation Considerations 5](#__RefHeading___Toc23777_1576096024)

[1.4. Reference Documents 5](#__RefHeading___Toc23779_1576096024)

[2. Design Overview 7](#__RefHeading___Toc23781_1576096024)

[2.1. Current State – Tc Integration with External Application - 7](#__RefHeading___Toc23783_1576096024)

[2.2. Key Assumptions 7](#__RefHeading___Toc23785_1576096024)

[2.3. Interface/API Definitions 7](#__RefHeading___Toc23787_1576096024)

[2.4. Classes Entities 8](#__RefHeading___Toc23789_1576096024)

[2.5. Classess Interaction flow 10](#__RefHeading___Toc23791_1576096024)

[2.5.1. Key Classes and Relationships 11](#__RefHeading___Toc23793_1576096024)

[2.6. Scalability and Capacity Planning 12](#__RefHeading___Toc23795_1576096024)

[2.7. Reusable Components 12](#__RefHeading___Toc23797_1576096024)

[2.8. Logging 13](#__RefHeading___Toc23799_1576096024)

[3. Design Overview 14](#__RefHeading___Toc23801_1576096024)

[3.1. Solution Description 14](#__RefHeading___Toc23803_1576096024)

[3.2. Agora Teamcenter Gateway Components and End to End Technical flow Explanation 15](#__RefHeading___Toc23805_1576096024)

[3.2.1. External Application – Provisioning System 15](#__RefHeading___Toc23807_1576096024)

[3.2.1.1. User Provisioning: 15](#__RefHeading___Toc23809_1576096024)

[3.2.1.2. Jammex: 15](#__RefHeading___Toc23811_1576096024)

[3.2.2. Agora Environment: 16](#__RefHeading___Toc23813_1576096024)

[3.2.2.1. Agora Teamcenter Gateway: 16](#__RefHeading___Toc23815_1576096024)

[3.2.2.2. Reception and Routing of the Request 16](#__RefHeading___Toc23817_1576096024)

[3.2.2.3. Interaction with Microservices 16](#__RefHeading___Toc23819_1576096024)

[3.2.2.4. Communication with AWS and Teamcenter 16](#__RefHeading___Toc23821_1576096024)

[3.2.2.5. Generation of the Response 16](#__RefHeading___Toc23823_1576096024)

[3.2.2.6. Response to Jammex 17](#__RefHeading___Toc23825_1576096024)

[3.2.3. AWS Environment: 17](#__RefHeading___Toc23827_1576096024)

[3.2.4. Agora Teamcenter: 17](#__RefHeading___Toc23829_1576096024)

[3.3. Data Flow Summary 17](#__RefHeading___Toc23831_1576096024)

[4. Agora Teamcenter Gateway - Microservices 18](#__RefHeading___Toc23833_1576096024)

[4.1. Microservices for fetching Metadata 18](#__RefHeading___Toc23835_1576096024)

[4.1.1. AGORA\_Tc\_Part\_GetTcStatus 18](#__RefHeading___Toc23837_1576096024)

[4.1.2. AGORA\_Tc\_Part\_GetLatestRev 19](#__RefHeading___Toc23839_1576096024)

[4.1.3. AGORA\_Tc\_Part\_ GetPartName 21](#__RefHeading___Toc23841_1576096024)

[4.2. Microservices for fetching CAD Data 22](#__RefHeading___Toc23843_1576096024)

[4.2.1. AGORA\_Tc\_Design\_Get3DModel 22](#__RefHeading___Toc23845_1576096024)

[4.2.2. AGORA\_Tc\_Design\_ GetJTFIles 24](#__RefHeading___Toc23847_1576096024)

[4.2.3. AGORA\_Tc\_Design\_Get2DDrawingFile 26](#__RefHeading___Toc23849_1576096024)

[4.3. Errors and Exception Handling 27](#__RefHeading___Toc23851_1576096024)

[4.4. Business Continuity and Disaster Recovery 28](#__RefHeading___Toc23853_1576096024)

[4.5. Security Handling 28](#__RefHeading___Toc23855_1576096024)

[4.6. Environment Details 28](#__RefHeading___Toc23857_1576096024)

1. **Introduction**

## Purpose

The technical design and solution implementation document provides the technical specifications for the design and the solution implementation details of the Agora Teamcenter integration with external applications in two modes 1) Realtime and 2) Publish/Subscribe Model.

This document is also created as part of the initial architecture review and the solution implementation details before the development of the proof of concept.

This document also serves as a blueprint for the development and maintenance of Agora Teamcenter Integration Microservices. It should be modified at any time to reflect approved changes to the integration between Agora Teamcenter and external applications.

The architecture is designed for scalability, security, and real-time processing. By leveraging AWS for cloud queuing and combining it with a microservice-based gateway, the solution is well-suited for robust integration between Teamcenter and external applications.

## Scope

|  |
| --- |
| **Description** |
| * Problem Statement – As of now, the Agora Teamcenter is integrated with MCAD applications CREO, SolidWorks and intended to integrate with NX as well. But it is also intended to integrate with external applications like Jammex, User Provisioning Systems and A-Teamcenter to enable data flow between Agora, Jammex, A-Teamcenter and User Provisioning Systems. * Scope   + In Scope: Web App for Agora Teamcenter Integration   + Out of Scope: Any Development out of this requirement * MVP: An MVP typically focuses on simple working Application which demonstrate the most critical business requirements, rather than attempting to develop the entire process right from the start. The Developed MVP addresses key processes of sending Request, connecting to Teamcenter, Fetching required details(result) from Teamcenter using Agora Teamcenter Gateway application. By starting with a bare-bones version, Teamcenter developers can iterate, refine, and enhance the solution based on real-world usage, feedback, business needs (helps in minimizing risks, reducing development costs, and improving the chances of success). |

Apart from the above activities, any other activity is out of scope for this development.

## Technical Design & Solution Implementation Considerations

The following areas are considered for the technical design and solution implementation.

* The detailed steps for each microservice, including request handling, data processing, and response generation, are outlined in the use case document, providing a clear roadmap for development.
* As this is a greenfield implementation, the gateway architecture, protocols, and integration points will be established from scratch, ensuring alignment with the AGORA environment’s overall design.
* The solution will be developed independently without the need to modify or integrate with any existing systems, allowing for maximum flexibility in the design and implementation process.
* The gateway will be built using the latest stable version of the Java Spring Framework and other core technologies, ensuring modern standards and long-term support in the production environment.
* Sample part metadata and design information provided by the Business will be used to test the microservices in various environments to ensure functionality and performance meet the defined requirements.
* All pre-requisites, dependencies, and configurations for the AGORA Teamcenter Gateway’s microservices will be identified and fulfilled to ensure smooth operation in development, testing, and production environments.
* The gateway will be designed with future scalability in mind, allowing for seamless extensions and integrations as the AGORA environment evolves and new requirements emerge.
* A comprehensive testing strategy will be employed, including unit testing, integration testing, and end-to-end testing across all environments, to validate the gateway’s functionality, performance, and security before production deployment.
* The classes and the design/implementation team shall have access only to the necessary components
* The various steps for these classes and use cases are detailed in the use case document.
* Best practices used by the industry in the PLM.
* All the configurations/customization must be stored in the code Repository like GIT or SVN.

## Reference Documents

The reader is strongly recommended to read the use case document referenced below for the detailed descriptions of various tasks and details. This document must be read in conjunction with the use case document to ensure the complete information is provided.

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Document Name | Link | Comment |
| 1 | Amazon Simple Queueing Services |  |  |
| 2 | Request Respone details for both User Provitioning and JAMMEX |  |  |

1. **Design Overview**

## Current State – Tc Integration with External Application -

This solution will be developed from scratch, given the absence of any pre-existing system.

## Key Assumptions

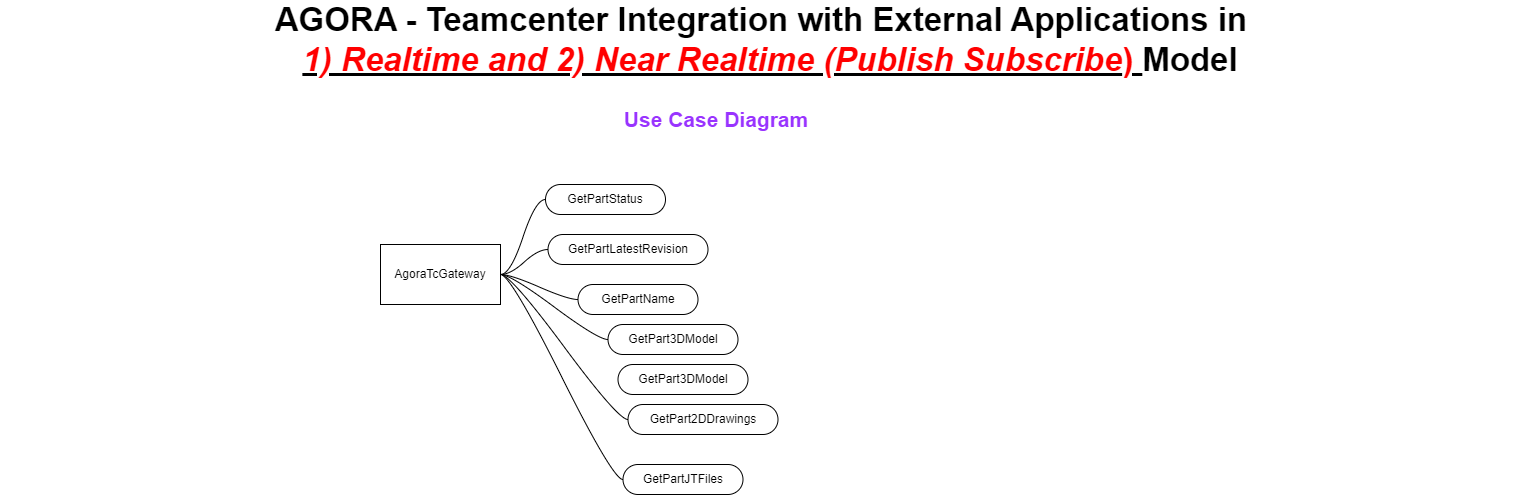
Below are the key assumptions for the proposed solution for “Tc Integration with External Application -”.

* The UserProvisioning Requests uses Amazon Simple Queuing Services
* A service Account will be created and used for all Gateway to Agora Tc communications.
* SSO or HTTPS: Security concerns should address single sign-on (SSO) and secure communication (HTTPS).
* VLAN/SUBNET: Consider network segmentation for security and performance.
* Open Source Usage: The architecture leverages open-source technologies like Tomcat and Spring Framework.
* Deployment: Ensure the components are correctly deployed and maintained.
* Pre-Requisites: Requirements for the setup, such as SSL configuration, appropriate AWS services, and network settings.

## Interface/API Definitions

An API document will be shared to the consumers that list out all the Microservices. Following are the OOTB API’s planned to be designed and developed in the base release.

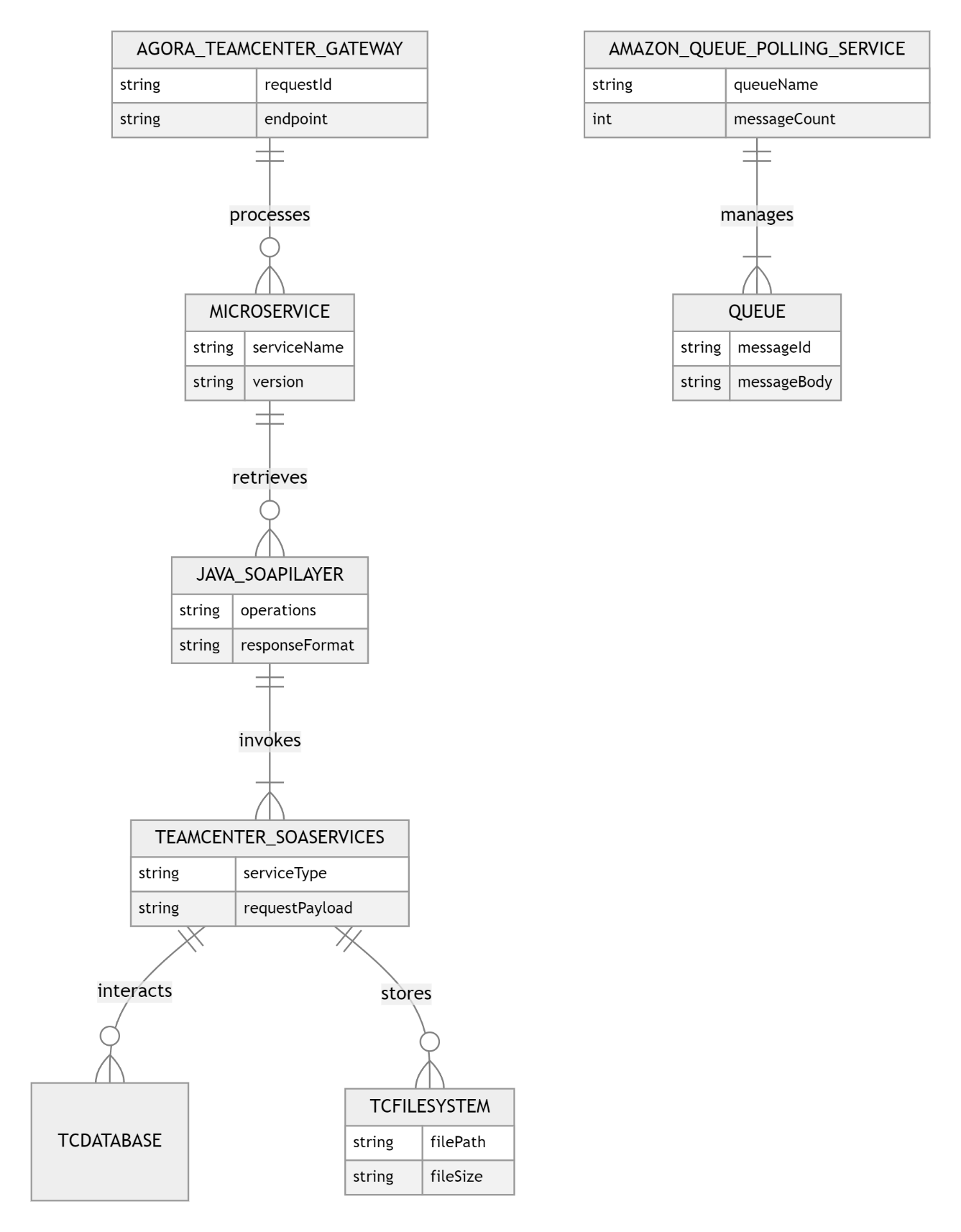
The following diagram shows the OOTB use cases developed.



* *Get3DModel*
* Get2DDrawing
* GetJTFiles
* GetTechSpecificationsDocument
* GetGeometry
* GetOwner
* GetPrefereedVendor
* GetMaterialFinish
* GetReleaseStatus
* GetCAGECODE
* GetReleaseDate

## Classes Entities

The following diagram details the various Classes designed as entities, involved in the Agora Gateway Integration of the proposed Solution.

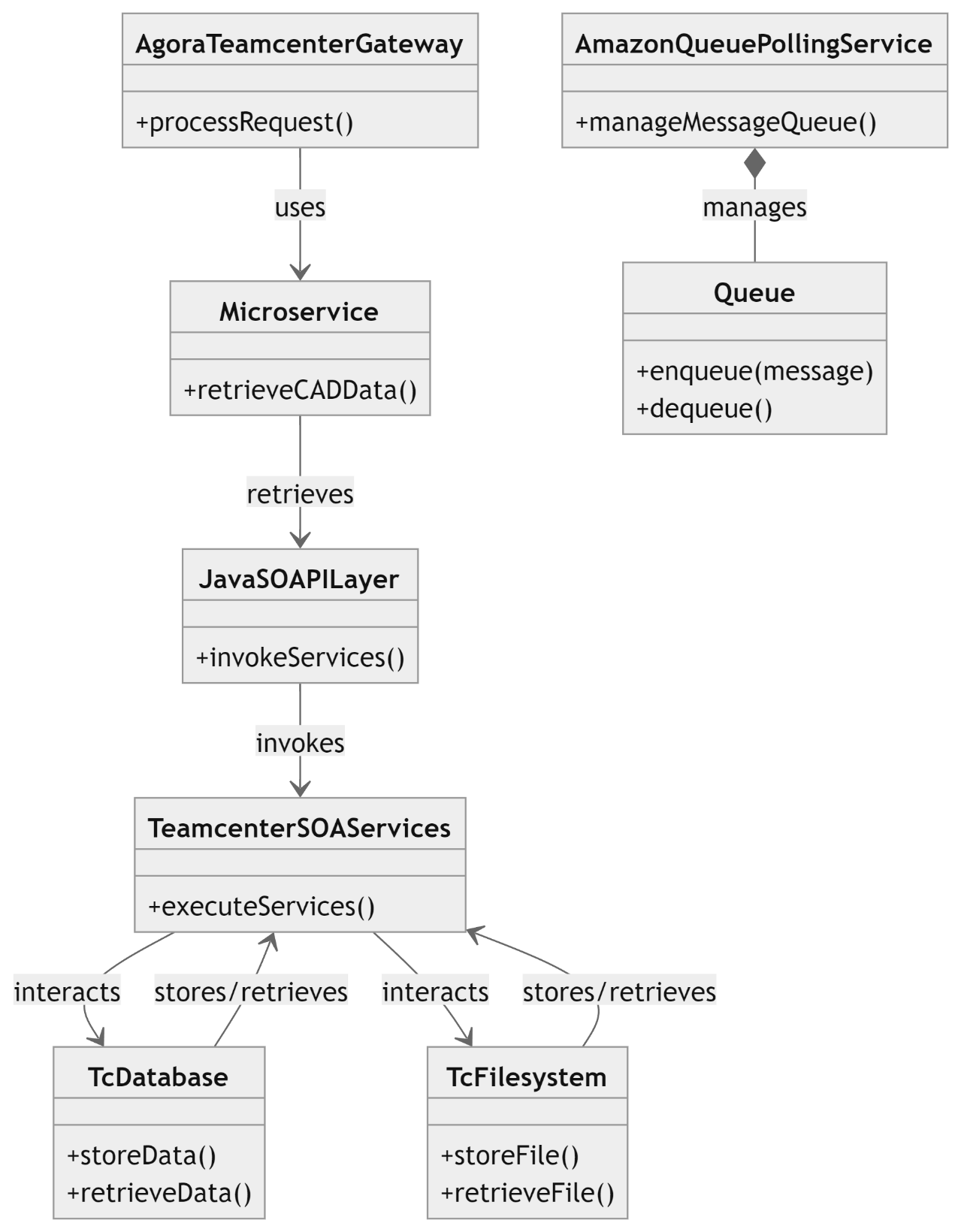


The following table details the high level Classes involved in this Agora Gateway Integration. As per the coding standards, it is recommeded to design the Classes to the lowest possible entity. Also it is designed to encapsulate all the related attributes/methods for each and every Classes involved in the Integration Application.

|  |  |
| --- | --- |
| **Entity** | **Description** |
| Agora Teamcenter Gateway | The AgoraTeamcenterGateway is an integration layer that facilitates communication between Jammex and Teamcenter, managing data exchange and routing requests. It coordinates interactions with AWS services and microservices for efficient processing. |
| Microservice | A Microservice is a modular component that performs specific functions and communicates through APIs. It enhances scalability and maintainability by encapsulating distinct tasks within separate services. |
| JavaSOAPILayer | The JavaSOAPILayer provides a standardized interface for interacting with SOAP-based web services using Java, converting requests and responses between SOAP and internal application logic. |
| TeamcenterSOAServices | TeamcenterSOAServices are SOA-based services from Teamcenter that expose functionalities such as part metadata retrieval through standardized interfaces, enabling integration with external systems. |
| AmazonQueue PollingService | The AmazonQueuePollingService monitors and processes messages from AWS SQS queues, facilitating asynchronous task processing and communication within the AgoraTeamcenterGateway. |
| Queue | A Queue in the AgoraTeamcenterGateway context refers to an AWS SQS that holds and manages messages related to tasks like user provisioning, ensuring orderly and reliable task processing. |
| TcDatabase | TcDatabase is Teamcenter's database system for storing and managing data related to parts, design information, and configurations, providing a structured repository for enterprise resources. |
| TcFilesystem | TcFilesystem is Teamcenter's file storage system for managing design files such as 3D models and technical specifications, ensuring efficient storage, retrieval, and organization of design information. |

## Classess Interaction flow

The following class diagram illustrates the key components of the AGORA Teamcenter Gateway architecture, highlighting the relationships between the gateway, microservices, AWS queue management, and Teamcenter SOA services. Each class and its interactions are clearly defined to support the flow of requests, processing, and data storage within the system.



## Key Classes and Relationships

|  |  |  |  |
| --- | --- | --- | --- |
| **Classes** | **Attribute** | **Method** | **Relationships** |
| AgoraTeamcenterGateway | None | +processRequest(request: HTTPRequest) | * Composes: Microservice * Uses: AmazonQueuePollingService * Uses: JavaSOAPILayer |
| Microservice | name: String | +getMetadata(partID: String)  +getCADData(partID: String) | Associated with: JavaSOAPILayer |
| JavaSOAPILayer | None | +invokeSOAPService(service: String, request: SOAPRequest) | Uses: TeamcenterSOAServices  Used by: Microservice |
| TeamcenterSOAServices | serviceName: String | +executeService(request: SOAPRequest) | Communicates with: TcDatabase  Communicates with: TcFilesystem |
| AmazonQueuePollingService | queueName: String | +pollQueue(queueName: String) | Composes: Queue (aggregation) |
| Queue | queueName: String | +enqueue(message:JSONRequest)  +dequeue(): JSONRequest | Aggregated by: AmazonQueuePollingService |
| TcDatabase | None | +storeMetadata(metadata: Metadata)  +retrieveMetadata(partID: String) | Accessed by: TeamcenterSOAServices |
| TcFilesystem | None | +storeCADData(data: CADData)  +retrieveCADData(partID: String) | Accessed by: TeamcenterSOAServices |

## Scalability and Capacity Planning

N/A.

## Reusable Components

Teamcenter SOA Api’s

Teamcenter built-in Functionalities

## Logging

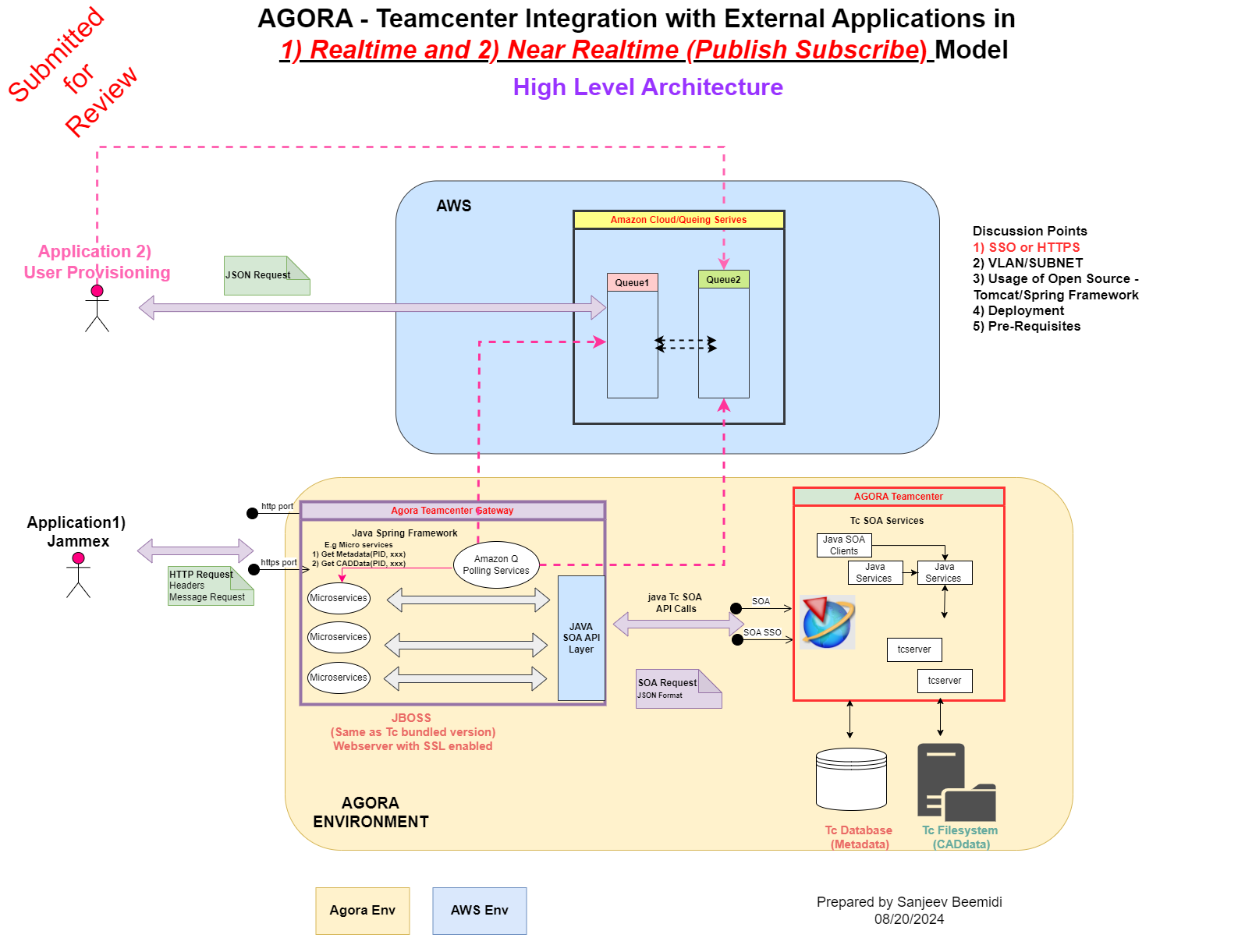
The following standard logging levels should be matched in the solution.

|  |  |
| --- | --- |
| **Log Level** | **Description** |
| **TRACE** | Verbose Log Entry, Very Detailed Information |
| **DEBUG** | Used to log debug information useful for debug malfunctioning Classes, but not considered business event information |
| **INFO** | Informational message tied to specific business event being performed. For example, “Work Item received”, “Write-off performed”, etc. |
| **WARN** | Message to notify about exceptional, but non-breaking business case or something out of ordinary, but not stopping automation from continuing |
| **ERROR** | Message about error during logic execution. In most cases such error would require Classes to start over or stop gracefully |
| **FATAL** | Unrecoverable error requiring Classes to stop. Example, configuration was not loaded. Send notifications and make fixing a priority |

1. **Design Overview**

## Solution Description

The following diagram details the high-level architecture of the Agora Teamcenter Gateway that integrates Agora Teamcenter with external applications.



This diagram represents the high-level architecture of AGORA's Teamcenter integration with external applications using a Realtime (Jammex) and Near-Realtime (Publish/Subscribe) model. The integration leverages Amazon Web Services (AWS) for cloud queuing services, microservices for handling business logic, and a gateway for interfacing with Teamcenter. The architecture also addresses key points such as SSO/HTTPS, VLAN/SUBNET, open-source usage, deployment, and prerequisites.

## Agora Teamcenter Gateway Components and End to End Technical flow Explanation

## External Application – Provisioning System

## User Provisioning:

1. It is understood that the user provisioning system is currently publishing the provisioning request to Amazon Simple Queuing Service (SQS). The published JSON object contains all the necessary information to provision the user in to Teamcenter application.
2. The request is submitted in to a “REQUEST QUEUE” and the response is received from a “RESPONSE QUEUE”
3. SQS is configured to retain the requests/response for a configurable number of hours before expiring.
4. The User Provisioning System prepares a JSON request and submit the JSON request object in to SQS.
5. AGORA Teamcenter Gateway contains Amazon Queue polling services, it will process the request and update the response to Amazon Response Queue.
6. It is assumed that the request JSON contains Teamcenter Organization information such as username, password, Default Group, Role, mail Address, etc., for creating user in Teamcenter.
7. User Provisioning is a near Realtime model, which will be run at specified time for a day. Once request is processed in Teamcenter using AGORA’s Teamcenter gateway, execution status and result will be updated to Amazon Queue.
8. users can find the response to their request in their respective system using Amazon Queue Response.
9. AGORA Teamcenter Gateway will creates Response in Response queue for each processed request.

## Jammex:

1. Jammex, targeting external users, initiates interaction with the AGORA Teamcenter Gateway by sending an HTTP or HTTPS request that may include headers or message content, with the data format and any additional information, such as part ID, being crucial for the gateway's processing.
2. Jammex is a Realtime model, it gives Realtime part status/information from Teamcenter to the user.
3. The requests sent by Jammex are typically HTTP or HTTPS-based, possibly following a RESTful approach. The data within the request must align with the expected format to ensure correct processing by the gateway.
4. Upon receiving the request, the gateway validates and routes it to the appropriate microservice or component for processing, which may involve interaction with Teamcenter. Finally, the gateway compiles and sends the response back to Jammex securely.

## Agora Environment:

## Agora Teamcenter Gateway:

1. The AGORA Teamcenter Gateway, based on the Java Spring Framework, acts as an intermediary between Jammex and the AGORA environment, including Teamcenter services. The gateway is designed to process and route requests, ensuring seamless communication and interaction between external applications and Teamcenter.

## Reception and Routing of the Request

1. Request Validation: Upon receiving an HTTP or HTTPS request from Jammex, the gateway first validates the incoming data, ensuring that it meets the required format and includes all necessary information, such as the part ID.
2. Routing: After validation, the gateway routes the request to the appropriate microservice or internal component within the AGORA Teamcenter Gateway.

## Interaction with Microservices

1. Microservices are Deployed within JBOSS (SSL-enabled web server) to handle business logic. These microservices communicate with Teamcenter via the gateway.
2. Microservice Routing: The gateway directs the request to the relevant microservice within the AGORA Teamcenter gateway . Each microservice is tailored to handle a specific business function, such as data processing, metadata management, or real-time part status updates. Etc,
3. Based on business use cases different Microservices will be defined. Example fetching Metadata details from Teamcenter like., Release status, Latest Revision, Part name and Fetching CAD Data like, 3D Model, JT Drawing, 2D Drawings.
4. SOA API Layer Communication: Communication between the gateway and microservices is managed through the JAVA SOA API Layer, ensuring standardized interaction and data flow. Interfaces between microservices and Teamcenter using SOA requests in JSON format.

## Communication with AWS and Teamcenter

1. AWS Queue Handling: For user provisioning, the AWS Queue will be utilized, running daily.
2. Teamcenter Interaction: When the request involves Teamcenter operations, the gateway communicates with Teamcenter SOA Services. These services are responsible for interacting with the Teamcenter Database and Filesystem to retrieve or update information.

## Generation of the Response

1. Compiling Results: After the microservices or Teamcenter operations have processed the request, the gateway compiles the results.
2. Response Formatting: The compiled results are then formatted according to the initial request specifications, typically in JSON format.
3. Secure Delivery: The gateway ensures the secure delivery of the response back to Jammex, maintaining data integrity and confidentiality.

## Response to Jammex

1. Synchronous Response: For synchronous requests, Jammex receives the requested data or confirmation of successful processing directly.
2. Asynchronous Response: For asynchronous requests, a status update or reference ID may be provided, enabling Jammex to track the request’s progress.

## AWS Environment:

* 1. Amazon Cloud Queuing Services:

1. Queue1 and Queue2: Handle message passing between Agora Teamcenter Gateway and external services.
2. The queues ensure reliable communication for real-time and near-real-time services.
3. Amazon Q polling service within Agora Teamcenter Gateway, will get requests in Request queue (Queue1), Processed in Teamcenter then status will updated back to Response Queue(Queue2).

## Agora Teamcenter:

1. Teamcenter SOA Services: Consists of SOA Clients, SOA Services, and Java Services.
2. The services interface with both the Teamcenter Database (Metadata) and the Teamcenter Filesystem (CAD data).
3. The gateway interacts with Teamcenter via SOA SSO, enabling secure communication.

## Data Flow Summary

1. It is assumed that User Provisioning is for Internal Users, sends a JSON request for creating users in Teamcenter which will be Queued in AWS. It will be run at specified time for a day by Amazon Q Polling services from AGORA Environment. Once Processed in Teamcenter by AGORA Teamcenter Gateway, results will be published back to Queues in AWS .
2. Jammex sends HTTP requests, which are routed to the Agora Environment. It is used to get Part metadata and CAD Data from AGORA Teamcenter in Realtime.
3. Requests are processed by microservices in the Agora Gateway Environment and results are forwarded to JAMMEX or AWS queues(based on request). AGORA Environment has AGORA Teamcenter Gateway which act as Intermediate/Bride between External and AGORA Teamcenter.
4. The AGORA Teamcenter gateway either receive request from JAMMEX or polls queues for tasks, processes them, and interfaces with Teamcenter through SOA API calls.
5. Teamcenter performs operations on the database and filesystem based on the processed requests.
6. **Agora Teamcenter Gateway - Microservices**

## Microservices for fetching Metadata

Microservices will be developed within the AGORA Teamcenter Gateway to retrieve essential metadata for parts, including information on the owner, preferred vendor, material finish, and release status, as well as design information such as 3D models, 2D drawings, JT files, technical specification documents, and geometry. High availability and quick response times will be prioritized in the design of these microservices, supporting real-time data needs.

An explanation of the input, output, and error handling for a few sample Part metadata and design information microservices has been provided in the following sections:

## AGORA\_Tc\_Part\_GetTcStatus

Get the Part metadata (Status) details from the AGORA Teamcenter

**Pre-Conditions**

A part must exist in AGORA Teamcenter

**Requirements:**

|  |  |
| --- | --- |
| AGORATC\_INTEG\_1.0 (AGILITY\_ID) | The AGORA Teamcenter Gateway shall provide the Teamcenter Part status Details for the search procedure requested by JAMMEX |
| AM023.01.JM.06 | The JAMMEX system API shall return a Part Status detail response. |

**Post-Conditions:**

A detail JSON file with all the mapped data is returned including the links to downloadable files.

**Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | jia/api/agora/revisionstatus/<part\_id> |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | A detail JSON response is provided based upon the mapped fields with the JAMMEX internal schema. Example response below:  {  "schema": "agora",  "part\_id": "<part\_id>",  "latest\_revision\_status": {  "revision\_id": "rev\_003",  "status": "Released",  "status\_description": "This revision has been approved and released for production.",  "attributes": {  "approved\_by": "Jane Doe",  "approval\_date": "2024-08-10T14:30:00.000-04:00",  "change\_description": "Updated with new material specifications.",  "release\_level": "Production"  },  "metadata": {  "created": "2024-07-20T10:15:00.000-04:00",  "last\_modified\_by": "John Smith",  "created\_by": "Jane Doe",  "last\_modified": "2024-08-10T14:45:00.000-04:00"  }  }  } |
| 404 | {"error": "No record exists with Id."+ <record\_id>} |
| 500 | {"error": "An error occurred while getting model details for the Record ID :” + record\_id + “ Please try again later. If the problem persists, please contact the JAMMEX Administrator."} |

## AGORA\_Tc\_Part\_GetLatestRev

Get the Part metadata (Latest Revision) details from the AGORA Teamcenter

**Pre-Conditions**

A part must exist in AGORA Teamcenter

**Requirements:**

|  |  |
| --- | --- |
| AGORATC\_INTEG\_1.0 (AGILITY\_ID) | The AGORA Teamcenter Gateway shall provide the Teamcenter LatestRevision Details for the search procedure requested by JAMMEX |
| AM023.01.JM.06 | The JAMMEX system API shall return a Part’s Latest Rev detail response. |

**Post-Conditions:**

A detail JSON file with all the mapped data is returned including the links to downloadable files.

**Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | jia/api/agora/latestrevision/<part\_id> |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | A detail JSON response is provided based upon the mapped fields with the JAMMEX internal schema. Example response below:  {  "schema": "agora",  "part\_id": "<part\_id>",  "latest\_revision": {  "revision\_id": "rev\_001",  "revision\_status": "Released",  "revision\_description": "Initial release of the part",  "attributes": {  "owner": "John Doe",  "preferred\_vendor": "Vendor A",  "material\_finish": "Anodized Aluminum",  "release\_status": "Released"  },  "metadata": {  "created": "2023-08-15T12:45:00.000-04:00",  "last\_modified\_by": "Jane Smith",  "created\_by": "John Doe",  "last\_modified": "2023-08-20T09:20:00.000-04:00"  },  "files": [  {  "path": "https://20.42.84.12:8080/agora/api/download/part/rev\_001/files/design\_3d\_model.stp",  "filename": "design\_3d\_model.stp",  "size": "25000",  "field": "3d\_model",  "mime\_type": "application/stp",  "id": "design\_3d\_model"  },  {  "path": "https://20.42.84.12:8080/agora/api/download/part/rev\_001/files/drawing.pdf",  "filename": "drawing.pdf",  "size": "102400",  "field": "2d\_drawing",  "mime\_type": "application/pdf",  "id": "drawing"  }  ]  }  } |
| 404 | {"error": "No record exists with Id."+ <record\_id>} |
| 500 | {"error": "An error occurred while getting part details for the Record ID :” + record\_id + “ Please try again later. If the problem persists, please contact the JAMMEX/Teamcenteter Administrator."} |

## AGORA\_Tc\_Part\_ GetPartName

Get the Part metadata (Part Name) details from the AGORA Teamcenter

**Pre-Conditions**

A part must exist in AGORA Teamcenter

**Requirements:**

|  |  |
| --- | --- |
| AGORATC\_INTEG\_1.0 (AGILITY\_ID) | The AGORA Teamcenter Gateway shall provide the Teamcenter PartName Details for the search procedure requested by JAMMEX |
| AM023.01.JM.06 | The JAMMEX system API shall return a PartName detail response. |

**Post-Conditions:**

A detail JSON file with all the mapped data is returned including the links to downloadable files.

**Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | jia/api/agora/partname/<part\_id> |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | A detail JSON response is provided based upon the mapped fields with the JAMMEX internal schema. Example response below:  {  "schema": "agora",  "part\_id": "<part\_id>",  "part\_name": {  "name": "Gear Housing Assembly",  "description": "Housing assembly for the primary gear mechanism.",  "category": "Assemblies",  "attributes": {  "owner": "Jane Doe",  "preferred\_vendor": "Vendor B",  "material": "Aluminum",  "release\_status": "Released"  },  "metadata": {  "created": "2023-07-10T10:30:00.000-04:00",  "last\_modified\_by": "John Smith",  "created\_by": "Jane Doe",  "last\_modified": "2023-08-01T08:00:00.000-04:00"  }  }  } |
| 404 | {"error": "No record exists with Id."+ <record\_id>} |
| 500 | {"error": "An error occurred while getting part details for the Record ID :” + record\_id + “ Please try again later. If the problem persists, please contact the JAMMEX/Teamcenteter Administrator."} |

## Microservices for fetching CAD Data

## AGORA\_Tc\_Design\_Get3DModel

Get the Design metadata (3D Model) details from the AGORA Teamcenter

**Pre-Conditions**

A part must exist in AGORA Teamcenter

**Requirements:**

|  |  |
| --- | --- |
| AGORATC\_INTEG\_1.0 (AGILITY\_ID) | The AGORA Teamcenter Gateway shall provide the Teamcenter Part’s 3D Model Details for the search procedure requested by JAMMEX |
| AM023.01.JM.06 | The JAMMEX system API shall return a 3D Model detail response. |

**Post-Conditions:**

A detail JSON file with all the mapped data is returned including the links to downloadable files.

**Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | jia/api/agora/3dmodel/<part\_id> |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | A detail JSON response is provided based upon the mapped fields with the JAMMEX internal schema. Example response below:  {  "schema": "agora",  "part\_id": "<part\_id>",  "3d\_model": {  "model\_id": "model\_456",  "filename": "gear\_housing.stp",  "description": "3D model of the gear housing assembly.",  "attributes": {  "material": "Aluminum",  "scale": "1:1",  "units": "millimeters"  },  "metadata": {  "created": "2024-07-20T14:30:00.000-04:00",  "last\_modified\_by": "John Smith",  "created\_by": "Jane Doe",  "last\_modified": "2024-08-01T10:00:00.000-04:00"  },  "file": {  "path": "https://20.42.84.12:8080/agora/api/download/part/3dmodel/model\_456/gear\_housing.stp",  "filename": "gear\_housing.stp",  "size": "204800",  "mime\_type": "application/stp"  }  }  } |
| 404 | {"error": "No record exists with Id."+ <record\_id>} |
| 500 | {"error": "An error occurred while getting part details for the Record ID :” + record\_id + “ Please try again later. If the problem persists, please contact the JAMMEX/Teamcenteter Administrator."} |

## AGORA\_Tc\_Design\_ GetJTFIles

Get the Design metadata (JT FIles) details from the AGORA Teamcenter

**Pre-Conditions**

A part must exist in AGORA Teamcenter

**Requirements:**

|  |  |
| --- | --- |
| AGORATC\_INTEG\_1.0 (AGILITY\_ID) | The AGORA Teamcenter Gateway shall provide the Teamcenter Part’s JT Files Details for the search procedure requested by JAMMEX |
| AM023.01.JM.06 | The JAMMEX system API shall return a JT FIles detail response. |

**Post-Conditions:**

A detail JSON file with all the mapped data is returned including the links to downloadable files.

**Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | jia/api/agora/jtfiles/<part\_id> |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | A detail JSON response is provided based upon the mapped fields with the JAMMEX internal schema. Example response below:  {  "schema": "agora",  "part\_id": "<part\_id>",  "jt\_files": [  {  "file\_id": "jt\_001",  "filename": "housing\_assembly.jt",  "description": "JT file for the housing assembly.",  "attributes": {  "file\_size": "150000",  "units": "millimeters",  "version": "10.5"  },  "metadata": {  "created": "2024-07-25T09:45:00.000-04:00",  "last\_modified\_by": "Jane Doe",  "created\_by": "John Smith",  "last\_modified": "2024-08-01T11:30:00.000-04:00"  },  "file": {  "path": "https://20.42.84.12:8080/agora/api/download/part/jtfiles/jt\_001/housing\_assembly.jt",  "filename": "housing\_assembly.jt",  "mime\_type": "application/jt"  }  },  {  "file\_id": "jt\_002",  "filename": "gear.jt",  "description": "JT file for the gear component.",  "attributes": {  "file\_size": "120000",  "units": "millimeters",  "version": "10.5"  },  "metadata": {  "created": "2024-07-26T14:20:00.000-04:00",  "last\_modified\_by": "Jane Doe",  "created\_by": "John Smith",  "last\_modified": "2024-08-02T10:10:00.000-04:00"  },  "file": {  "path": "https://20.42.84.12:8080/agora/api/download/part/jtfiles/jt\_002/gear.jt",  "filename": "gear.jt",  "mime\_type": "application/jt"  }  }  ]  } |
| 404 | {"error": "No record exists with Id."+ <record\_id>} |
| 500 | {"error": "An error occurred while getting part details for the Record ID :” + record\_id + “ Please try again later. If the problem persists, please contact the JAMMEX/Teamcenteter Administrator."} |

## AGORA\_Tc\_Design\_Get2DDrawingFile

Get the Design metadata (2D Drawing) details from the AGORA Teamcenter

**Pre-Conditions**

A part must exist in AGORA Teamcenter

**Requirements:**

|  |  |
| --- | --- |
| AGORATC\_INTEG\_1.0 (AGILITY\_ID) | The AGORA Teamcenter Gateway shall provide the Teamcenter Part’s 2D Drawing Details for the search procedure requested by JAMMEX |
| AM023.01.JM.06 | The JAMMEX system API shall return a Model detail response. |

**Post-Conditions:**

A detail JSON file with all the mapped data is returned including the links to downloadable files.

**Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | jia/api/agora/2ddrawing/<part\_id> |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| 200 | A detail JSON response is provided based upon the mapped fields with the JAMMEX internal schema. Example response below:  {  "schema": "agora",  "part\_id": "<part\_id>",  "2d\_drawing\_file": {  "file\_id": "2d\_123",  "filename": "gear\_housing\_drawing.pdf",  "description": "2D drawing of the gear housing assembly.",  "attributes": {  "drawing\_number": "GH-001",  "revision": "B",  "scale": "1:1",  "units": "millimeters",  "sheet\_size": "A3"  },  "metadata": {  "created": "2024-07-28T12:00:00.000-04:00",  "last\_modified\_by": "Jane Doe",  "created\_by": "John Smith",  "last\_modified": "2024-08-03T14:45:00.000-04:00"  },  "file": {  "path": "https://20.42.84.12:8080/agora/api/download/part/2ddrawing/2d\_123/gear\_housing\_drawing.pdf",  "filename": "gear\_housing\_drawing.pdf",  "mime\_type": "application/pdf"  }  }  } |
| 404 | {"error": "No record exists with Id."+ <record\_id>} |
| 500 | {"error": "An error occurred while getting part details for the Record ID :” + record\_id + “ Please try again later. If the problem persists, please contact the JAMMEX/Teamcenteter Administrator."} |

## 

## Errors and Exception Handling

Exceptions handled by the gateway can primarily be classified into system and business exceptions. The gateway under discussion handles each error type in a standardized manner.

In general, if the microservices encounters any unexpected scenario, the microservices will generate an exception message and inform the same to the external application. The microservices will generate an exception message with the details of the error stack trace.

**Business Exceptions**

All known business exceptions are accounted for within the solution design. If an unexpected scenario is found, the microservices will log an exception in the logfile for that record and pass the traceback to the calling application.

Any unexpected business scenarios the microservices encounters will stop working and continue for the next request.

|  |  |  |
| --- | --- | --- |
| **Exception No.** | **Exception Status for Reporting** | **Exception Description** |
| **1** | Fail | Application Connectivity |
| **2** | Skip | Agora Part ID is invalid |
| **3** | Success | Agora ID and all input parameters are correct and all the responses are sent back without any issues. |

For any business exception, then:

1. Action taken by Microservices: Error captured in the Log/Report for the business exceptions
2. Log/Report for Exception: The Exception message for the business scenario will be preserved within the Log Files.
3. Manual Action required:

a. Teamcenter Admin: Manual review of Logs/Report recommended if a business exception frequently occurs

b. Microservices Controller: None

## Business Continuity and Disaster Recovery

N/A.

## Security Handling

None of the information is shared outside of this context.

## Environment Details

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Environment** | **FQDN/Machine Name (Full Qualified domain name)** | **IP Detail** |
| 1 | DEV |  |  |
|  |  |
|  |  |
| 1 | UAT |  |  |
|  |  |
|  |  |
|  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |