ICT Project Guidance

Technical Integrations via ETL

Author:

Sky Sigal

## Description

This document describes how to develop an integration to a target system using Extraction, Transformation, Load (ETL) approach directly to its data store.

## Synopsis

Integrations via ETL are not the preferred approaches because they bypass system logic, including in-system Authorisation, Authentication, Auditing, logical modelling and validation. However, the approach is required when the target system either does not have APIs or they are insufficient to meet desired outcomes.

## Contents

[Description 1](#_Toc147235799)

[Synopsis 1](#_Toc147235800)

[Contents 2](#_Toc147235801)

[Introduction 4](#_Toc147235802)

[Issue 4](#_Toc147235803)

[Resolution 4](#_Toc147235804)

[ETL 4](#_Toc147235805)

[Views 4](#_Toc147235806)

[Integration View 6](#_Toc147235807)

[Delivery View 8](#_Toc147235808)

[Approval 8](#_Toc147235809)

[Purpose & Scope Synopsis 8](#_Toc147235810)

[Business System Owner Approval 8](#_Toc147235811)

[Discovery 8](#_Toc147235812)

[System Support Specialists 8](#_Toc147235813)

[Database Schema 8](#_Toc147235814)

[Database Server Name & Database Name 8](#_Toc147235815)

[Database View(s) Design Document 9](#_Toc147235816)

[Dependencies 9](#_Toc147235817)

[Service Account 9](#_Toc147235818)

[Domain Groups 9](#_Toc147235819)

[Service Agent Device 10](#_Toc147235820)

[Service Agent Device Directory 10](#_Toc147235821)

[Deployment Agent Device 10](#_Toc147235822)

[Deployment Agent 11](#_Toc147235823)

[Database Login 11](#_Toc147235824)

[Development 11](#_Toc147235825)

[DDL Code 11](#_Toc147235826)

[Database Server Login 11](#_Toc147235827)

[Database Server Authorisation 12](#_Toc147235828)

[Database User 12](#_Toc147235829)

[View(s) 12](#_Toc147235830)

[Permissions 12](#_Toc147235831)

[Agent & Packaging Development 12](#_Toc147235832)

[Agent 13](#_Toc147235833)

[Agent Installation Package 13](#_Toc147235834)

[Cloud Deployment Pipeline 14](#_Toc147235835)

[Intranet Deployment Pipeline 14](#_Toc147235836)

[Response Target 14](#_Toc147235837)

[Deployment 14](#_Toc147235838)

[Well known Agent Deployment Package Source 14](#_Toc147235839)

[Operations View 15](#_Toc147235840)

[Instruction Package 15](#_Toc147235841)

[Recoverable Error Logging 15](#_Toc147235842)

[Successful Queries 15](#_Toc147235843)

[Monitoring 15](#_Toc147235844)

[Summary 16](#_Toc147235845)

[Appendices 17](#_Toc147235846)

[Appendix A - Document Information 17](#_Toc147235847)

[Images 17](#_Toc147235848)

[Tables 17](#_Toc147235849)

[References 17](#_Toc147235850)

[Review Distribution 17](#_Toc147235851)

[Audience 17](#_Toc147235852)

[Diagrams 17](#_Toc147235853)

[Terms 17](#_Toc147235854)

[Appendix B – Assumptions 18](#_Toc147235855)

[Relational Database 18](#_Toc147235856)

[SQL Server 18](#_Toc147235857)

[Appendix C – Issues with Integrating with UI APIs 18](#_Toc147235858)

# Introduction

Integration between a system and 3rd party systems may be required to import data from them.

The preferable approach is to integrate over Application Programming Interfaces (APIs), because they can be expected to while limiting requests to Authenticated and Authorised systems, keeping audit records of operations, validate request arguments, and only requiring current and commonly available development skills to implement.

## Issue

While APIs are preferred, they are not always available for use. Immaturely developed or older systems developed in an era before APIs were commonly developed may not provide integration APIs[[1]](#footnote-2).

In other cases, while APIs have been developed for User Interface (UI) needs, they may not be useful for integration purposes. This is because APIs for Integration purposes are distinct from the purpose of APIs for User Interfaces (UI), the difference posing several issues, as listed in the appendices.

## Resolution

The preferable approach is for the remote system to add APIs for integration.

When this is not a viable option, integrations reliant on an automated “Extract, Transform, Load” (ETL) approach is appropriate.

## ETL

Essentially, ETL involves installing an agent that is provided scope constrained read only access to Views within the target system’s database for the purpose of extracting data by queries triggered by scheduling or a received signal.

## Views

When the target system’s database can be accessed and modified without technical or contractual impact, use-case specific Views[[2]](#footnote-3) are the preferred means to isolate each integration from others.

Note:  
While direct access to a database is technically simpler, the development and use of Views to constrain access to only the require tables and records is the preferred means to meet security and compliance obligations.

Note:  
While stored procedures can also be developed using constrained Views is preferred: they diminish the complexity of the development and ongoing quality assurance required from an external team.

# Integration View

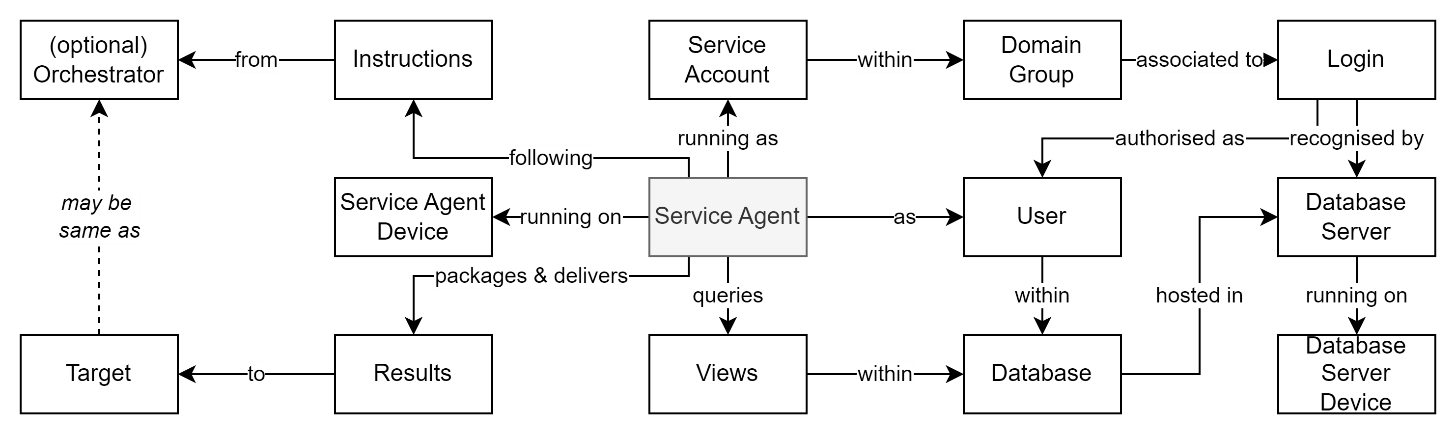


Figure 1: ETL Integration Components

At a high level, an ETL integration relies on the following components working together.

Table 1: ETL Integration Components (HL)

##### [Target] Database

: the target system’s database, mounted within:

##### [Target] Database Server

: the target system’s database.

##### Service Agent’s Service Account

: a Network Domain Service issued Service Account, that is a member of a Network Domain Service developed Account Group, which has been associated to an Integrated Login[[3]](#footnote-4) in the target Database Server[[4]](#footnote-5), and is a User in the remote database, and has been granted access to one or more Views within it.

##### Service Agent

: an executable running on the Service Agent Device.

##### Service Agent Device

: the device to which the Service Agent is deployed in the same private network as the *Database Server*, so that it can make queries as the *Service Agent’s Service Account* to the target Database over a non-http port (e.g., 1433) over a private network.

##### Account Group

: a Network Domain Service developed group of Service Accounts.

**Important:**For maintainability quality outcomes, access to the database is granted to the Database User who’s Database Server Login is associated to a Domain Service Account Group, and not a Service Account within it.

Note:  
It is common practice to reuse the same Service Account to access all target system environments, a member of multiple Groups, each one permitted to a different target environment.

##### Query Instructions

: the Agent follows instructions to query the Views and deliver the query response to a target destination.

Note:  
In immature scenarios the instructions can be hard coded into the agent and a redeployment of the agent would be required if there changes to the instructions are required. A preferred option is for the Agent to be deployed with Instructions, or receive Instructions from an remote Orchestrator.

##### Orchestrator

: an (optional) provider of Instructions to the Service Agent. The instructions may be delivered when the Service Agent is deployed, or the Service Agent may call the Orchestrator, using either an outward bound[[5]](#footnote-6) WebSocket or Long Polling based connection[[6]](#footnote-7).

##### Query Results [Package]

: the Agent may optionally package one or more queries into a single result package, and deliver it to a target destination.

##### Target

: a location where Query Results are delivered. Depending on the use case, the location may be:

* an API endpoint, belonging to a service in the same local network or a different remote network, that accepts packaged a push of the Query Results.  
  (e.g., the orchestrator itself, or a cloud storage API to which the Service Agent has been granted access).
* an accessible drop folder, noting that this legacy approach is no longer considered best-practice).

# Delivery View

Delivering the above connection involves a number of phases, starting with a getting Approval, then a Discovery phase, followed by a Design, Development and Delivery phase.

## Permission

A prerequisite to connecting to a service is getting prior approval from the owners of the system providing the business service.

### Purpose & Scope Synopsis

A description of the business purpose and scope of information requested, with a synopsis of the privacy impact should be developed to assist with their deciding to grant approval.

### Business System Owner Approval

The above document is used to obtain approval to access the data.   
This is a prerequisite to proceeding on to discovery of the technical aspects of establishing a connection.

## Discovery

One approval has been obtained and evidenced, discovery can take place.

### System Support Specialists

A prerequisite is contacting the specialist resources maintaining the target system’s datastores in both non-prod data environments and prod data environments.  
From them the following can be established:

#### Database Schema

The schema[[7]](#footnote-8) of the target database is required early, for analysis and developing an understanding of the minimum set of tables required to be queried, their attributes, and how many Views will be required to be developed to isolate this information for external access.

#### Database Server Name & Database Name

The target system’s database is required, as is the name of the database server.

If following best practice of using one database server per target environment, there will be several (e.g., BT, ST, UT, PP, PR), within different domain networks (e.g., an ST and PROD network), the access of which is managed by the network’s directory service (e.g., [Active] Active Directory).

The information will be required to correctly develop connection strings between Service Agents and the database server.

Note:  
the name of the database may or may not be the same in every environment. This information will be required to be know, to provide another attribute of the connection string required by the service agent described later.

### Database View(s) Design Document

A deliverable from the discovery phase is a document describing the View(s) that are required.

Note:  
a View design may summarise or show combined rows from multiple tables, may limit the number of columns or rows returned and relabel them to better describe their contents.

The design document can contain a high-level logical description and/or high-level physical description of the desired attributes. It is rare that there is any merit to describing attribute types, etc. as these are easily discoverable by the implementor.

The document is provided to the specialist resources in charge of the target system’s database.

But the implementation of the design as code is dependent on other deliverables being developed first.

## Dependencies

### Service Account

Service Accounts are required to be commissioned within each domain controller service of the required network (e.g., both the ST, PROD domains).

It is best practice that each connection type between devices is provided its own service account.

Note:  
Organisation specific naming conventions may apply, to make it easier to determine in diagnostic logs which service invoked the connection, and where it was intending to connect to.

### Domain Groups

As multiple environments will exist (e.g., BT, DT, ST, UT, PP, PR) but will be created within a smaller number of network (e.g., an ST and PR domains), the same service account is used across multiple target system environments (e.g. one across BT, DT, ST, UT, and one across PP, PR).

Instead, access is controlled by the creation of Service Account Groups, one for each target environment (e.g.: BT, DT ST, UT. PP, PR), in the appropriate domain controller.

Hence Groups are required to be commissioned for each target database environment (e.g., BT, DT, ST, UT, PP, PR) within the domain service of the relevant networks (ST, PROD).

### Service Agent Device

As a cloud-based server-less based logic is not recommended[[8]](#footnote-9), a physical or virtual device will be required to deploy an Agent to.

Note:  
If using the recommend approach described above, of using a common service account, with access to multiple Groups, then one device per network will be sufficient.

### Service Agent Device Directory

A decision is required as to where on the (potentially shared) device the agent is to be deployed later.

The target installation directory will require at least two sub directories, one for executables (e.g., bin), and one for writing temporary, rolling, diagnostics text logs (e.g., logs).

Note:   
The directory will require read/write access from the service account under which the deployment automation runs.

### Deployment Agent Device

The deployment of the service agent to its intended device, and its subsequent configuration, is expected to meet current best practice, and use DevOps managed automation[[9]](#footnote-10).

For cloud-based devops automation an Automation Agent Device is required to commissioned within each network (ST, PP), so that a Deployment Agent can be then be deployed.

### Deployment Agent

Once a device is available in each network, a Deployment Agent is required to be installed and configured to connect and call back to its cloud-based automation orchestration service to await deployment instructions.

Note:  
Whether the Deployment Agent (and optionally the inclusion of the underlying Device) is developed and configured by hand or by a separate on-site/non-cloud-based deployment automation (e.g., via Puppet) is left as an implementation choice.

### Database Login

Each Database in each Environment will require a Login to be developed, associated to the User Group developed for that Environment.  
While this step can be done manually, it is recommended that it be done via automation, hence described in the next section.

## Development

### DDL Code

The changes to the target database can be done manually, but it is preferable that the changes are done via coded instruction, using the database server’s Database Definition Language (DDL).

Therefore, the development of new coded instructions is required to automate the following tasks:

* Development of a Database User associated to the Database Server Login that is associated to the database specific Domain Group.
* Creation of View(s) according to the provided View description documentation.
* Providing the database User with Permissions to the View(s).

The developed code is expected to be persisted in a code store and run over the target databases in each environment.

#### Database Server Login

A Login is required to be developed on the database server.

SQL Server offers two modes of authentication[[10]](#footnote-11) of a server Login.

Note:  
SQL Server Authentication, reliant on local storage of databases and subsequent transfer of confidential credentials over the wire is not to be used, in favour of using “Windows/Integrated Authentication”, which validates principal tokens and does not require the transfer of confidential information over the wire.

Windows Authentication permits the authentication of both local device or domain accounts or groups.

Enabling a domain group via windows authentication is recommended. This permits multiple users (e.g., system maintainers, service agents, deployment agents) to login to the database server – once they are added to the appropriate domain group (see above) that is associated to a specific database server.

#### Database Server Authorisation

The database server Login does not require being granted an extended set of rights: *public* access is expected to be sufficient for most use cases.

#### Database User

Whereas a Login permits *authentication* to a server, it does not automatically *authorise* access as a User to a specific database managed by the database server.

Therefore, the authenticated Database Server Login is required to registered first as a User[[11]](#footnote-12) in the target Database.

#### View(s)

The development of the Views is based on the design document provided earlier.

#### Permissions

Finally, the database User is then authorised/granted *select* rights to only the newly created case specific Views[[12]](#footnote-13).

### Agent & Packaging Development

The Service Agent may be custom developed or be by a vendor.

If in-house,

#### Agent

The Agent’s purpose is to query View(s) as a thread running as the domain user who has been added to the domain group that been mapped to the database server login, who is a User with appropriate rights to the View(s).

The Agent may be developed in a few ways:

* as a Windows service or Linux daemon, running in permanence in the background without a visible user interface, either:
  + triggering the request at intervals, or
  + upon receiving a signal to do so over a WebSocket or Long Polling based connection[[13]](#footnote-14).

If the agent has not been developed as a service / daemon, a scheduled task can be used to start a short-lived executable to invoke the query.

Considerations with this approach include:

* working to suppress the display of the console and/or user interface,
* as the scheduling mechanism is separate from the agent code and is part of the operating system it makes deployment require additional rights.

Note:  
The use of service accounts/daemons for running long standing executables in the background is a more preferred design to the use of scheduled tasks.

#### Agent Installation Package

The agent will require being packaged to facilitate its installation on a target device.

#### Cloud Deployment Pipeline

The deployment of the service agent installation package, its activation, and any related configuration should be performed via a cloud-service-based automation (e.g.: Azure Pipelines, etc.).

The pipeline will download the Agent Installation Package and instruct the Deployment Pipeline Agent to deploy and unpack it into the target (bin) directory of the Deployment Pipeline Agent Device, and configure it as needed.

Configuration to code per target database will include:

* Instructing it to use the network’s User Account when making queries.
* Instructing it to as to which Database Server and Database to use.
* Instructing it where to deliver query results to.
* Instructing it as to which credentials to use to access the delivery target.

### Intranet Deployment Pipeline

Any code required to deploy the Deployment Pipeline Service Agent via an on-prem deployment pipeline automation process (e.g., using Puppet) is beyond the scope of this document.

### Response Target

When the service agent makes queries to a View to which it has permissions, it packages the response, and delivers it to a target.

The target can be the Orchestrator’s API or drop folder created for the purpose.

Either way, the target needs to be developed if necessary and access enabled to the service account under which the service agent runs.

## Deployment

The final steps are installing the Service Agent that will be Instructed to package and return the responses from queries to the developed Views.

### Well known Agent Deployment Package Source

The deployment pipeline used to install the service agent has to retrieve the Agent installation package from somewhere.

If the agent is custom built, or this organisation has the source code, then the well-known source will probably be its code repository.

Note:  
If the agent is developed by a 3rd party vendor, then the well-known source will have to be decided.

The deployment is to executable folder (bin) under the previously chosen *Service Agent Device Directory.*

Important:  
The objective is that every time the deployment script is run that it retrieves the same version to idempotently deploy it - until configured to do otherwise.

# Operations View

If the Service Agent is controlled by an Orchestrator, then Operations can occur as follows.

## Instruction Package

The Service Agent download an instructions package from the Orchestrator which instructs it to query the View.

If the instructions include instructions to query other schema objects than the agreed views, it will result in a recoverable error.

## Recoverable Error Logging

If the Service Agent cannot query the View, it is expected to develop diagnostic trace logs in the (logs) directory of the *Service agent Device Directory*.

## Successful Queries

Successful queries will be package and delivered to the configured Target.

## Monitoring

Monitoring can be developed as required. Consider

* observing system Errors
* observing that the Target is getting regular result packages.

# Summary

A diagram summarizing the main aspects of the connection can be viewed below.

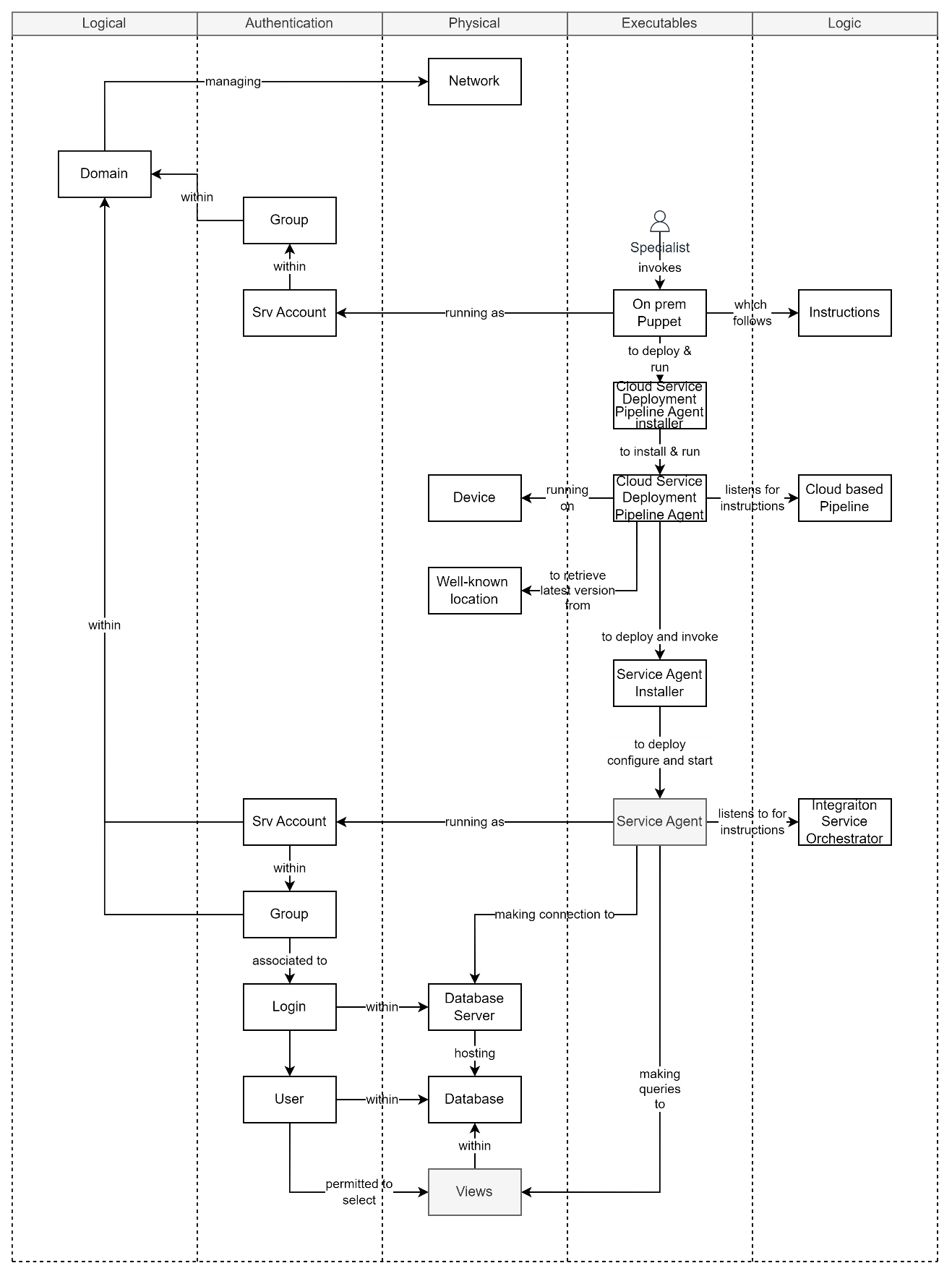


Figure 2: Logical Relationship Diagram

Appendices

Appendix A - Document Information

### Images

[Figure 2: Logical Relationship Diagram 15](#_Toc147156916)

### Tables

[Table 1: ETL Integration Components (HL) 5](#_Toc147235199)

### References

**There are no sources in the current document.**

### Review Distribution

The document was distributed for review as below:

|  |  |
| --- | --- |
| Identity | Notes |
| Sandy Britain, Enterprise Architect |  |
| Matt Duguid, DevOps Specialist |  |
|  |  |

### Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

### Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

### Terms

Refer to the project’s Glossary.

##### Term

: the meaning.

Appendix B – Assumptions

Some assumptions have been made in this document.

#### Relational Database

In this document it is assumed that the target database is a relational database, on the assumption that being without usable APIs it may have been develop before non-relational datastore were widely used in an enterprise environment.

#### SQL Server

It is also assumed that, based on probability, the database server used is SQL Server.

Appendix C – Issues with Integrating with UI APIs

Issues with using APIs intended for UIs for integration purposes may have one or more of the following issues:

* even with access to the current Users list of Tenancies they are members of, there is no guarantee they are members of all target system Tenancies,
* even if the current User is a member of all Tenancies, the extraction process would involve making multiple request to each all tenancies,
* it is common practice that a user be constrained to a single tenancy at a time so the operations cannot be sped up by performing them in parallel via multithreading,
* even if the current User is a member of all Tenancies, it is common for a User to be able to only query for their own data,
* even if the current User has sufficient rights in each tenancy to retrieve a summary report on all Users within the Tenancy, the summary information may not be sufficiently detailed for purpose,
* auditing of the operation to switch tenancies, and/or respond to the query, would be expected, so the overall duration of a single operation would be expected to be constrained by one or more I/O speeds.
* On target systems with thousands of tenancies with hundreds of users each, it is possible that the overall duration of the operation -- rotating through querying all tenancies and/or all users one at a time – may not be achievable within an afterwork period that would not affect business users.[[14]](#footnote-15)

Appendix C – Issues with Integrations by Database

The following issues with integrations directly to a database are relevant to ETL operations:

* Inserting Data bypasses logic in the Logic tier of an application.
  + Logic in a database is not considered best practice. It duplicates logic that should exist in a higher tier, and unless mirroring exactly the same logic introduce subtle and hard to discover issues.
* Extracting Data from databases directly bypasses authentication, authorisation and session operation auditing logic that exists in the logic tier.
  + The use of Views – as recommended in this document -- mitigate access to information outside the scope of an agreement, it cannot enforce requests using more fine-grain authorisation logic.
  + Database server logging is not suitable for auditability purposes as it of a temporary nature, while being separate from the system’s permanent auditing information.

1. APIs for integration are for a purpose distinct from servicing service clients. [↑](#footnote-ref-2)
2. [CREATE VIEW (Transact-SQL) - SQL Server | Microsoft Learn](https://learn.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql?view=sql-server-ver16) [↑](#footnote-ref-3)
3. Commonly referred to as “Windows Authentication”. [↑](#footnote-ref-4)
4. As per the Appendices, the database is assumed to be SQL Server. [↑](#footnote-ref-5)
5. This removes a dependency on a firewall rule change. [↑](#footnote-ref-6)
6. [Introduction to SignalR | Microsoft Learn](https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr) [↑](#footnote-ref-7)
7. [Retrieving database schema information - ADO.NET Provider for SQL Server | Microsoft Learn](https://learn.microsoft.com/en-us/sql/connect/ado-net/retrieving-database-schema-information?view=sql-server-ver16) [↑](#footnote-ref-8)
8. It would require the target network being misconfigured to permit the DB’s ports accessible over the www. [↑](#footnote-ref-9)
9. Preferably future focused, toward cloud automation as opposed to legacy, on-prem, focused. [↑](#footnote-ref-10)
10. [Choose an authentication mode - SQL Server | Microsoft Learn](https://learn.microsoft.com/en-us/sql/relational-databases/security/choose-an-authentication-mode?view=sql-server-ver16) [↑](#footnote-ref-11)
11. [CREATE USER (Transact-SQL) - SQL Server | Microsoft Learn](https://learn.microsoft.com/en-us/sql/t-sql/statements/create-user-transact-sql?view=sql-server-ver16) [↑](#footnote-ref-12)
12. [GRANT Object Permissions (Transact-SQL) - SQL Server | Microsoft Learn](https://learn.microsoft.com/en-us/sql/t-sql/statements/grant-object-permissions-transact-sql?view=sql-server-ver16) [↑](#footnote-ref-13)
13. [Introduction to SignalR | Microsoft Learn](https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr) [↑](#footnote-ref-14)
14. 5000 tenancies x 0.5seconds = 42 minutes. 100,000 users = 13.8 hours. [↑](#footnote-ref-15)