CTX-Manual-Intervention Deployment Plan

Contents

Versions

Document Revisions

The following revisions have been made to this document

Date	Revision	Notes
16/09/2019	0.1	First Draft
23/9/2019	1.0	First Release

Module Versions

This version of the CTX-Manual-Intervention deployment plan is relevant up to version 1.0 of the CTX-Manual-Intervention module.

Preface

About this Manual

This document provides a guide on how to deploy the CTX-Manual-Intervention module in your Cortex system.

Audience

This document is intended for those who require the use of CTX-Manual-Intervention module.

Related Material

Document
CTX-Manual-Intervention – User Guide
CTX-Manual-Intervention.studiopkg
CTX-Manual-Intervention-Update.sql

Abbreviations used in this Document

SQL Structured Query Language

DB Database

1 Requirements

This document details all the steps required to deploy the CTX-Manual-Intervention module.

Requirements:

- SQL Server Management Studio Access to the Cortex Database Server
- Minimum Cortex v6.4 installed on the Cortex Application Server
- Minimum SQL Server 2012 (version 11.0.7001.0) installed on the Cortex Database Server
- CTX-Logging Module installed on server
- CTX-Configuration-Store installed on server
- CTX-LivePortal-Artefacts (included with this studiopkg file).

2 Import CTX-Manual-Intervention

To deploy the CTX-Manual-Intervention module on your Cortex system, CTX-Manual-Intervention Studio Package needs to be imported on your Cortex system. To do this:

- Download the CTX-Manual-Intervention Studio Package
- Import the Studio Package in Cortex Gateway
- Ensure the relevant users have the required permissions in 'Studio Authorisation'

After this, all users in the authorised groups will be able to view and execute the subtasks.

3 Update Cortex-Logging Database

3.1 Overview

For the CTX-Manual-Intervention module to work, the Cortex-Logging database along with the schema must exist on the server where the Cortex databases exist and some minor (non-breaking) changes must be made to it. The following steps instruct you how to deploy the database and schema.

3.1.1 Partitioned Solution

If the Logging database has been installed as a partitioned database, the following steps are needed:

1. Run the 'usp_PART_ProcessLog' ALTER script
2. Run the 'CommitLog' ALTER script
3. Run the following SQL: exec usp_PART_CreatePartition
    a. Note that if the partitioned DB has already been deployed, the existing partitioned tables must be deleted and the data in the SYS_PART_ViewTables and SYS_PART_TableList must be cleared out

3.1.2 Non-Partitioned Solution

If the Logging database has been installed as a normal database, the following steps are needed:

4. Run the 'ProcessLog Recreate' DROP + CREATE script
5. Run the 'CommitLog' ALTER script

3.2 Update Database

1 Open Remote Desktop Connection to the Cortex database server or SQL Server Management Studio Remote Access

2 Copy the relevant scripts to the database server

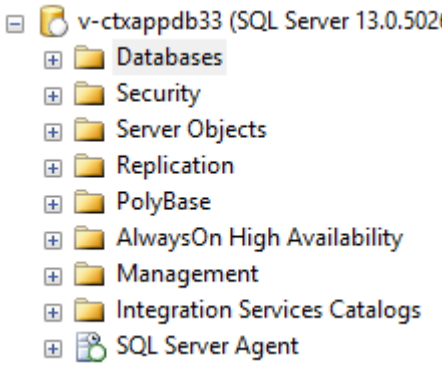
3 Open the scripts (as described in the **Overview** section) in SQL Server Management Server (SSMS) and connect to the DB engine where the query should be executed.

4 Select the Database Name containing the logging database in the top left dropdown

5 Before proceeding, ensure that the SQL Server Agent is running. This can be checked under Services > SQL Server Agent (MSSQLSERVER) for a Default SQL Server Instance or SQL Server Agent (<Instance Name>) for a Named SQL Server Instance

6 Run the scripts in turn. On the messages panel, you should see no errors

7 In the left-hand panel, click the plus to the left of 'Databases' to expand 'Databases'



8 Right click 'Databases' and click 'Refresh'.

9 Expand 'Cortex-Logging' (presuming the default Database Name was selected)

10 Expand 'Tables'

11 Expand 'Programmability' > 'Stored Procedures'

12 Validate the ProcessLog table(s) now have 2 new columns, UI and Email.

13 Validate the CommitLog Stored Procedure explicitly states all of the columns in every 'INSERT INTO ProcessLog' statement

14 Test the deployment in Cortex Gateway

3.3 Summary of Database Changes

3.3.1 ProcessLog Table

For the Manual Intervention initiative, 2 columns have been added to the ProcessLog Table

- UI (bit column) – indicates whether a UI Action is required in a process
- Email (bit column) – indicates whether a UI Action is required in a process

If it is a Partitioned solution, the stored procedure which creates all the partitioned ProcessLog tables (usp_PART_ProcessLog) should be updated. These tables will then be created once the usp_PART_CreatePartition procedure has been run.

3.3.2 CommitLogs Stored Procedure

Due to the change in the ProcessLog table, the Stored Procedure has been modified. Whenever a value is inserted into the ProcessLog table, the columns are now explicitly stated and they set both UI and Email to '0' (no action required).

Any updates to these columns are handled through a separate subtask instead of the CommitLog Stored Procedure.

4 Additional Files

4.1 LivePortal Components

The following sections cover the files needed for some of the UI components for the Manual Intervention Dashboard, the interactive LivePortal flow which allows the UI Escalations to be dealt with.

Item	Type	Server Destination (example)
ManualIntervention.js	Javascript File	C:\inetpub\Cortex\Cortex\Scripts
Tables	Folder (CSS)	C:\inetpub\Cortex\Cortex\Styles
breadcrumbs.css	CSS File	C:\inetpub\Cortex\Cortex\Styles\Intervention
button.css	CSS File	C:\inetpub\Cortex\Cortex\Styles\Intervention

Note that the destination for the files may be different based on the Cortex LivePortal Site configuration in IIS.

4.1.1 CSS Files

The Tables .zip file should be extracted and the folder copied into the Styles folder in the LivePortal IIS subdirectory. This is used to style the HTML Grid.

The Breadcrumbs.css file should be added to a new folder named ‘Intervention’ within the same subdirectory. This contains the CSS files for the breadcrumbs navigation.

4.1.2 JavaScript Files

ManualIntervention.js should be copied into the /Scripts/ directory within the LivePortal IIS subdirectory. This contains scripts to interact with the UI, for example selecting the clicked grid element, selecting the breadcrumbs navigation and the dashboard items displayed in the 1st UI page.

5 Configuration Store

5.1 Config Setup

To use the example flows and dashboards, the Cortex Configuration Store is needed. Once this has been deployed, the UI flow should be used to enter the following Config Parameters and the relevant Values:

Area	Customer	Environment	Param_Name
Manual-Intervention	NULL	NULL	Cortex-URL
Manual-Intervention	NULL	NULL	Database-Server
Manual-Intervention	NULL	NULL	Database-Name
Manual-Intervention	NULL	NULL	IMAP-Password
Manual-Intervention	NULL	NULL	IMAP-Port
Manual-Intervention	NULL	NULL	IMAP-Server
Manual-Intervention	NULL	NULL	IMAP-Username
Manual-Intervention	NULL	NULL	LivePortal-URL

Note that Environment should be provided for instances where more than one Cortex Server exists.

Database Name must point to the Logging DB used for Manual Intervention.

Cortex-URL should be the base URL used to trigger a REST request to the Flow API, and LivePortal-URL is used to open the relevant UI pages when an execution has been triggered.

As an example, if you would normally connect to https://myApp.myDomain.com/liveportal you would want the following values:

Cortex-URL: https://myApp.myDomain.com

LivePortal-URL:
https://myApp.myDomain.com/liveportal/SequenceHandling.aspx?execution=

Note that IP Addresses can also be used for these values.

6 Potential Issues

6.1 Logging Subtask – Partitioned vs Non-Partitioned

The Logging Database can be partitioned, which relies on UniqueIdentifier columns for the Primary Keys (normally this would be an integer). Depending on which version is deployed, the Logging Subtask may need to be modified.

The fix for this is to check the DB Query block in the logging subtask.

6.1.1 Non-Partitioned

```
DECLARE @ProcessLogID int,
                @StageLogID int,
                @EventLogID int

{{cl_Log-Main-Query}}

SELECT   @ProcessLogID as 'Process',
                @StageLogID as 'Stage',
                @EventLogID as 'Event'
```

6.1.2 Partitioned

```
DECLARE @ProcessLogID uniqueidentifier,
                @StageLogID uniqueidentifier,
                @EventLogID uniqueidentifier

{{cl_Log-Main-Query}}

SELECT   @ProcessLogID as 'Process',
                @StageLogID as 'Stage',
                @EventLogID as 'Event'
```

