CTX-State-Engine Deployment Plan

Contents

Versions

Document Revisions

The following revisions have been made to this document:

Date	Revision	Notes
08/02/2021	1.0	First release
09/11/2021	2.0	Second Release

Module Versions

This version of the CTX-State-Engine deployment plan is relevant up to version 2.0 of the CTX-State-Engine module.

Preface

About this Manual

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

Audience

This document is intended for those who require the use of CTX- State-Engine module.

Related Material

Document
CTX- State-Engine – User Guide
CTX- State-Engine.studiopkg
Cortex-State-Engine -Install.sql
Replication KeepAlive.sql
Add-Access-for-DBs
CTX_ConfigStore-Data-Setup_for_Cortex-State-Engine

Abbreviations used in this Document

SQL Structured Query Language

DB Database

1 Requirements

This document details all the steps required to deploy the CTX-State-Engine module.

Requirements:

- SQL Server Management Studio Access to the Cortex Database Server
- Minimum Cortex v6.5 installed on the Cortex Application Server
- Minimum SQL Server 2012 (version 11.0.7001.0) installed on the Cortex Database Server
- Deployment of the CTX-Task-Scheduler Module.
- Deployment of the CTX-Configuration-Store Module, including the SQL Cortex-ConfigStore database.
- Deployment of the CTX-Logging Module is required for the module to function correctly but can be deployed separately at any time and is not required to install the CTX-State-Engine Module.

2 Import CTX-State-Engine Flows

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

- Download the CTX- State-Engine 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 Deploy Cortex-State-Engine Database

3.1 Overview

For the CTX-State-Engine module to work, the Cortex-State-Engine database along with the schema must exist on the server where the Cortex databases exist. The following steps instruct you how to deploy the database and schema.

3.2 Create Single Site Database

1 Open 'Cortex-State-Engine -Install.sql' in SQL Server Management Server (SSMS) and connect to the DB engine where the query should be executed (this is where Cortex DBs are hosted).

2 Replace the SQL command variables as required:

```
:setvar CortexDBUser "domain\CTX_App_User_Service_Account"

:setvar DatabaseFilePath "C:\Cortex Databases"

:setvar DatabaseLogPath "C:\Cortex Databases"

:setvar Distribution_DataPath "C:\Cortex Databases\Distribution"

:setvar Distribution_LogPath "C:\Cortex Databases\Distribution"

:setvar InstanceName "LIVE_DBServerName\LIVE_InstanceName"

:setvar isReplicated "True"

:setvar MachineName " CortexDatabaseServer"

:setvar REPL_Admin_User "domain\CTX_SQL_Admin_User"

:setvar REPL_Working_Directory "C:\Cortex Databases\Replication
Data"

:setvar ResilientInstanceName "DR_DBServerName\DR_InstanceName"

:setvar DatabaseName "Cortex-State-Engine"

:setvar DefaultFilePrefix "Cortex-State-Engine"
```

Variable	Description
CortexDBUser	The Cortex Database Interface username on your Cortex system. Example: AD\CTX_CerberusDB
DatabaseFilePath	The directory to install the database Datefile to. Inside this directory there must be the folders <Database-Name>\Datafile, e.g. C:\Cortex Databases\Cortex-Logging\Datafile – note that the highlighted section should not be included in the variable.
DatabaseLogPath	The directory to install the database Logfile to. Inside this directory there must be the folders <Database-Name>\Logfile, e.g. C:\Cortex Databases\Cortex-

Variable	Description
	Logging\Logfile – note that the highlighted section should not be included in the variable.
Distribution_DataPath	The Distribution DB Datafile Path for the Db Server Instance. This must contain the following sub-folders: • Distribution ○ Datafile
Distribution_LogPath	The Distribution DB Logfile Path for the Db Server Instance. This must contain the following sub-folders: • distribution ○ Logfile
InstanceName	The name of the SQL Server or SQL Server Instance. If there are no named instances, then this should just be the server name or Machine Name.
isReplicated	"True" if the database is to be replicated, "False" otherwise for a Standalone installation
MachineName	The name of the database server on which the database is being installed
REPL_Admin_User	The administrator user on the replicated database (i.e. the other one, not the one where the database is being installed on now). In case of a standalone deployment, this parameter can be set to blank value.
REPL_Working_Directory	The directory for replication data on the replicated database (i.e. the other one, not the one where the database is being installed on now). In case of a standalone deployment, this parameter can be set to blank value.
ResilientInstanceName	Name of the server hosting replicated database (i.e. the other server, not the one where the database is being installed on now). In case of a standalone deployment, set this parameter to the same value specified for Instance Name. If there are no named instances, then this should just be the server name or Machine Name.
DatabaseName	The State Engine database name. It is advised to leave the default value 'Cortex-State-Engine'. Changing this value require updating the module flows/subtasks default values.
DefaultFilePrefix	To be prepended to filenames for search purposes

Example of a Standalone Implementation

```
:setvar CortexDBUser "LAB\CTX_SecureHub"
:setvar DatabaseFilePath "C:\Cortex Databases"
:setvar DatabaseLogPath "C:\Cortex Databases"
:setvar Distribution_DataPath ""
:setvar Distribution_LogPath ""
:setvar InstanceName "V-CTXDB17\DBLIVE"
:setvar isReplicated "false"
:setvar MachineName "CTXDB17"
:setvar REPL_Admin_User ""
:setvar REPL_Working_Directory ""
:setvar ResilientInstanceName "V-CTXDB17\DBLIVE"
:setvar DatabaseName "Cortex-State-Engine"
:setvar DefaultFilePrefix "Cortex-State-Engine"
```

Example of a Server without Named instance

```
:setvar CortexDBUser "LAB\CTX_SecureHub"
:setvar DatabaseFilePath "C:\Cortex Databases"
:setvar DatabaseLogPath "C:\Cortex Databases"
:setvar Distribution_DataPath "C:\Cortex Databases\Distribution"
:setvar Distribution_LogPath "C:\Cortex Databases\Distribution"
:setvar InstanceName "V-CTXDB17"
:setvar isReplicated "true"
:setvar MachineName "CTXDB17"
:setvar REPL_Admin_User "LAB\ctx_sql_admin"
:setvar REPL_Working_Directory "C:\Cortex Databases\Replication Data"
:setvar ResilientInstanceName "V-CTXDB18"
:setvar DatabaseName "Cortex-State-Engine"
:setvar DefaultFilePrefix "Cortex-State-Engine"
```

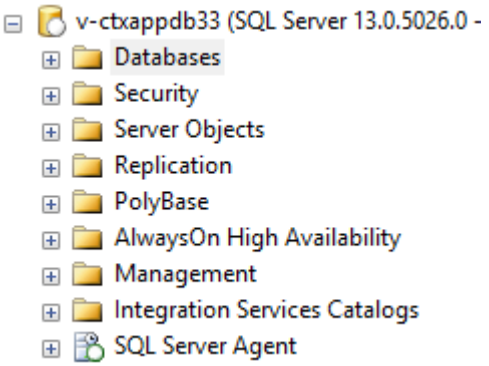
3 Before proceeding, ensure that the SQL Server Agent is running. This can be checked under Services > SQL Server Agent (MSSQLSERVER) for a Default SQL Server Instance or SQL Server Agent (<Instance Name>) for a Named SQL Server Instance.

4 Ensure that <Database-Name>\Datafile and <Database-Name>\Logfile folders exist within the DatabaseFilePath and DatabaseLogPath directories defined above respectively.

5 Click on ***Query*** -> ***SQLCMD Mode*** and execute the query

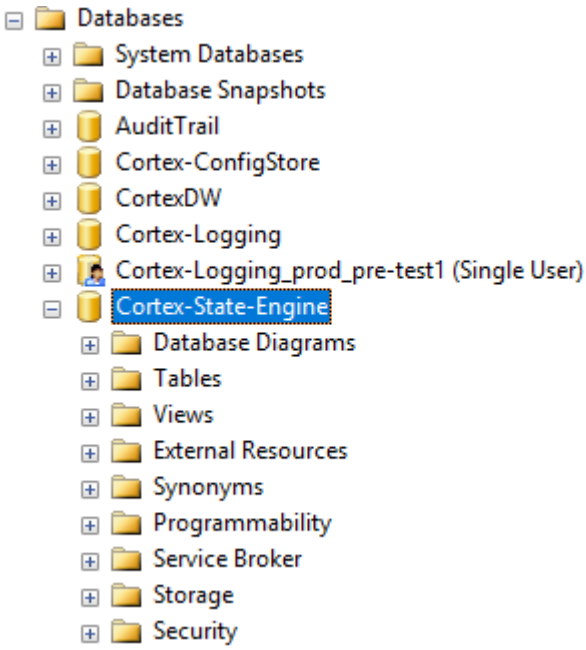
6 Wait for the messages panel at the bottom to inform the user that the script has finished executing.

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



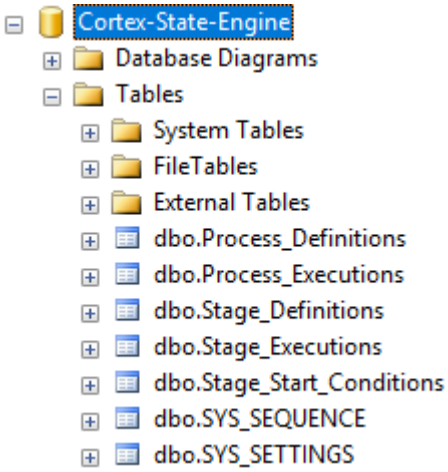
8 Right click ‘Databases’ and click ‘Refresh’.

9 Validate the ‘Cortex-State-Engine’ database has been created.



10 Expand ‘Cortex-State-Engine’ (presuming the default Database Name was selected)

11 Expand ‘Tables’, validate that the following has been installed correctly:



12 Expand ‘Programmability’ > ‘Stored Procedures’ and ‘Programmability’ > ‘Functions’ > ‘Scalar-Valued Functions’. validate that following has been installed correctly:

- Cortex-State-Engine
  - Database Diagrams
  - Tables
  - Views
  - External Resources
  - Synonyms
  - Programmability
    - Stored Procedures
      - System Stored Procedures
      - dbo.usp_ExecuteSQLStmt
      - dbo.usp_Get_Job_Status
      - dbo.usp_REPL_Add_Article
      - dbo.usp_REPL_Add_Publication
      - dbo.usp_REPL_Add_Subscription
      - dbo.usp_REPL_Create_LogReader_Job
      - dbo.usp_REPL_Create_Publication_Transactional
      - dbo.usp_REPL_Create_QueueReader_Job
      - dbo.usp_REPL_Create_Replicate_Job
      - dbo.usp_REPL_Create_Snapshot_Job
      - dbo.usp_REPL_CreateDefaultData
      - dbo.usp_REPL_Drop_Publication
      - dbo.usp_REPL_Remove_Replication
      - dbo.usp_REPL_Setup_Replication
      - dbo.usp_REPL_Start_Job
    - Functions
      - Table-valued Functions
      - Scalar-valued Functions
        - dbo.udf_Get_Parameter_Value
      - Aggregate Functions
      - System Functions
    - Database Triggers
    - Assemblies
    - Types
    - Rules
    - Defaults
    - Plan Guides
    - Sequences
  - Service Broker
  - Storage
  - Security

If a single site State Engine is required, then no further database setup is required, all the tables that are needed by the flows are present.

3.3 Create Replicated Database Platform

3.3.1 Set Up Replication

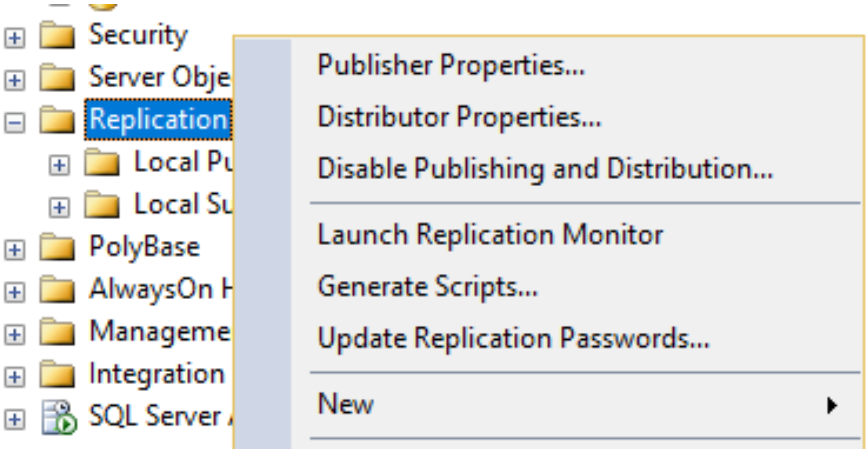
1. To begin with, carry out the steps in Section 3.2 on the resilient site, making sure to swap the resilient and local database server names around in the list of config variables.

   Now we have two separate database instances with the same schema, and we need to activate the replication between them. See below an example of Site-A and Site-B config side by side

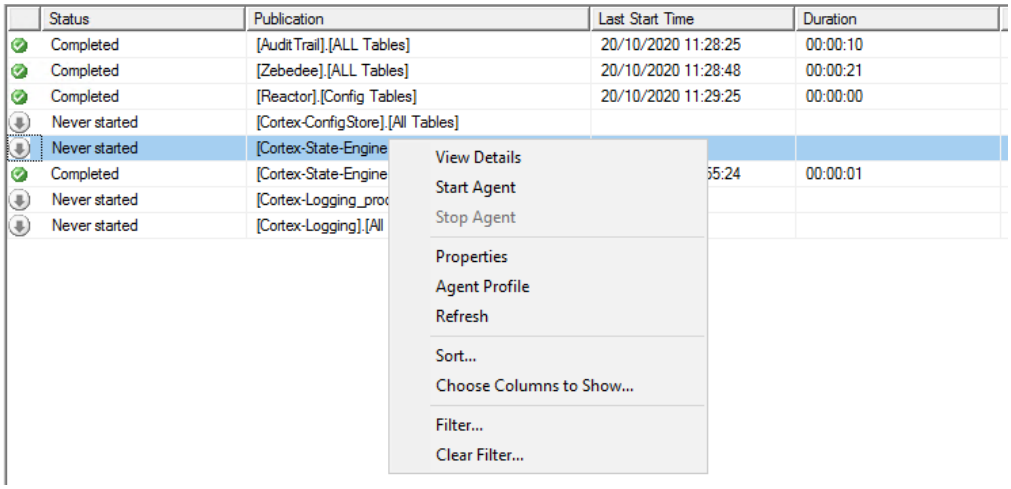
```
:setvar CortexDBUser "LAB\CTX_SecureHub"
:setvar DatabaseFilePath "C:\Cortex Databases"
:setvar DatabaseLogPath "C:\Cortex Databases"
:setvar Distribution_DataPath "C:\Cortex Databases\Distribution"
:setvar Distribution_LogPath "C:\Cortex Databases\Distribution"
:setvar InstanceName "V-CTXDB17\DBLIVE"
:setvar isReplicated "true"
:setvar MachineName "CTXDB17"
:setvar REPL_Admin_User "LAB\ctx_sql_admin"
:setvar REPL_Working_Directory "C:\Cortex Databases\Replication Data"
:setvar ResilientInstanceName "V-CTXDB18\DBDR"
:setvar DatabaseName "Cortex-State-Engine"
:setvar DefaultFilePrefix "Cortex-State-Engine"
```

```
:setvar CortexDBUser "LAB\CTX_SecureHub"
:setvar DatabaseFilePath "C:\Cortex Databases"
:setvar DatabaseLogPath "C:\Cortex Databases"
:setvar Distribution_DataPath "C:\Cortex Databases\Distribution"
:setvar Distribution_LogPath "C:\Cortex Databases\Distribution"
:setvar InstanceName "V-CTXDB18\DBDR"
:setvar isReplicated "true"
:setvar MachineName "CTXDB18"
:setvar REPL_Admin_User "LAB\ctx_sql_admin"
:setvar REPL_Working_Directory "C:\Cortex Databases\Replication Data"
:setvar ResilientInstanceName "V-CTXDB17\DBLIVE"
:setvar DatabaseName "Cortex-State-Engine"
:setvar DefaultFilePrefix "Cortex-State-Engine"
```

2. On the first site that was set up, carry out the following:
   a. Expand the stored procedures for the newly installed Cortex-State-Engine database
   b. Right click ‘REPL_Setup_Replication’
   c. Select ‘Execute Stored Procedure’. This will open the command to execute the stored procedure in a new query window and automatically execute it.
   d. Once this is complete, do the same for the stored procedure ‘REPL_Create_Publication_Transactional’
3. Perform the steps in step 2 on the resilient site.
4. Expand ‘Replication’ on either one of the two sites, and select ‘Launch Replication Monitor’



5. Navigate to the “Agents” tab
6. Right click the agent named Cortex-State-Engine (Assuming that was what the database was named) and select “Start Agent”.



3.3.2 Keep Replication Alive

Once replication has been implemented,

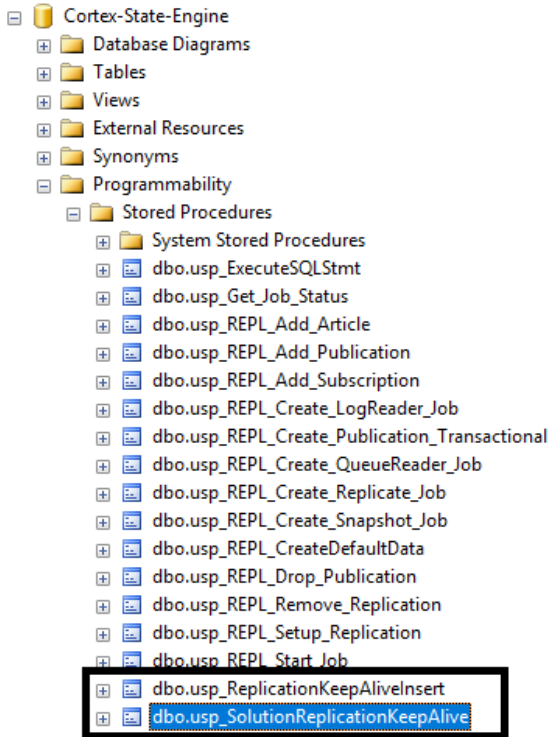
1. Open 'Replication KeepAlive.sql' in SQL Server Management Server (SSMS) and connect to the DB engine where the query should be executed (this is where Cortex DBs are hosted).
2. Replace the SQL command variables as required:

```
-- User Inputs:

:setvar LiveDatabaseServer 'V-CTXDB17'

:setvar ResilientDatabaseServer 'V-CTXDB18'

:setvar SolutionDatabasesExist 'True'

-- Type in the solution Database names here as comma-separated values (e.g. 'Cortex-ConfigStore,Cortex-Logging,CWT')

:setvar SolutionDatabaseNames 'Cortex-State-Engine'
```

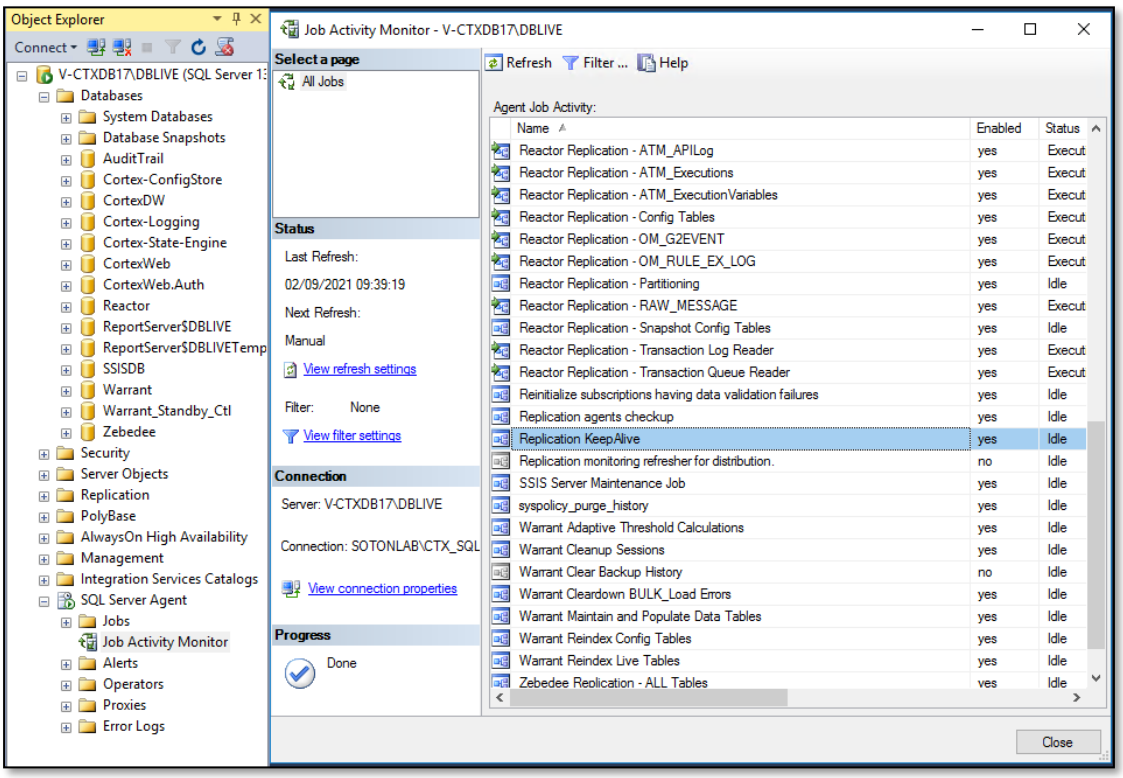
   a. Populate LiveDatabaseServer / ResilientDatabaseServer with DBServerName or DBServerName\InstanceName

3. This script will create required Stored procedures for Cortex-State-Engine DB and then add required automated Agent job steps to Replication-KeepAlive agent job. In case this job already exists for Reactor Database, it will just add an additional step.

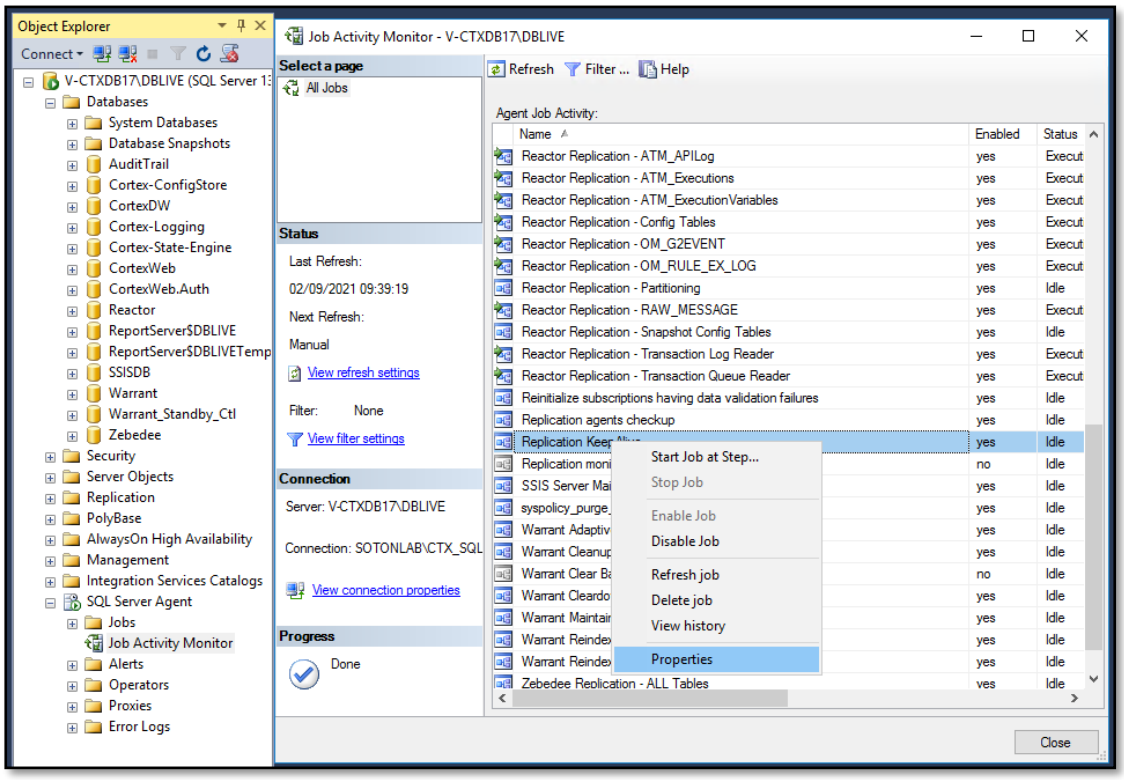
4 Check the DB for Stored procedures: Programmability-> Stored Procedures for 2 additional procedures:



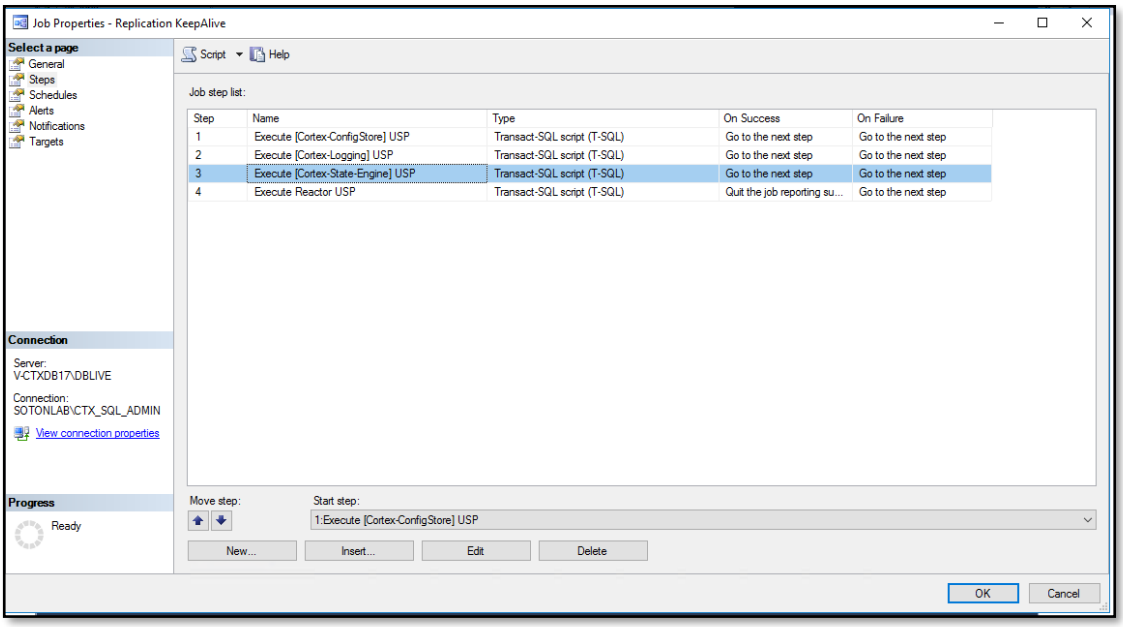
5 Verify the SQL Agent job by launching 'Job Activity Monitor' as shown below:



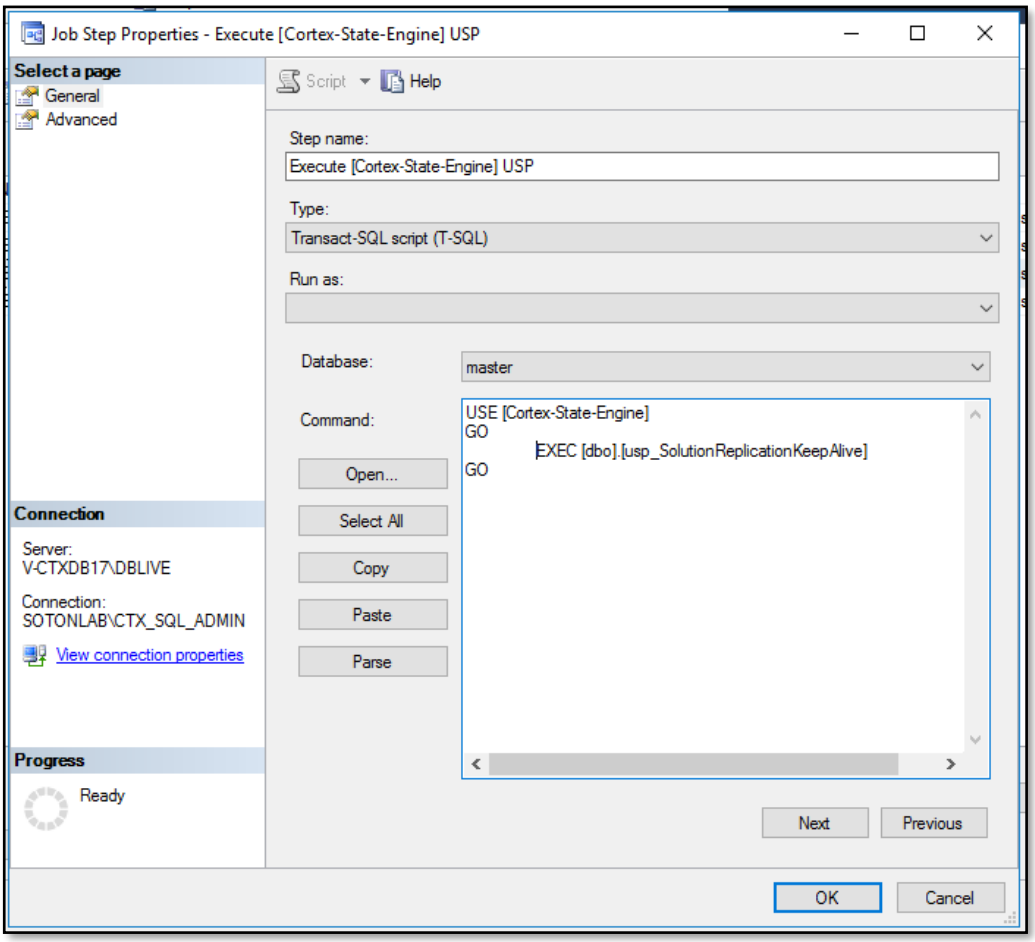
6 Right click on the job to view the properties:



7 Click on Step that contains Cortex-State-Engine:



8 Click on Edit button to view the selected step:



3.4 Add Access for Databases

The Cortex app server service user would require db_datareader, db_datawriter and db_executor access to the created DBs (LIVE and DR).

1 Copy the 'Add-Access-for-DBs' script to the Cortex database server

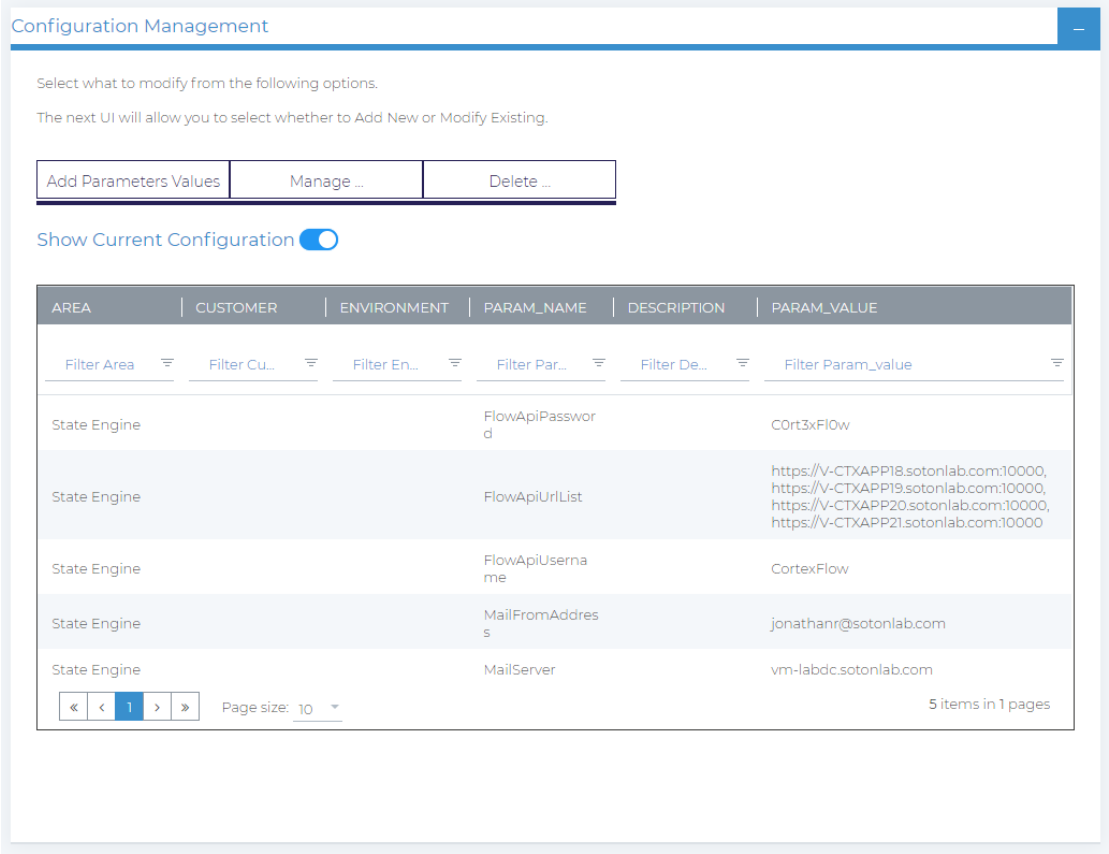
2 Open 'Add-Access-for-DBs' in SQL Server Management Server (SSMS) and connect to the DB engine where the query should be executed (this is where Cortex DBs are hosted).

3 Change the below DB user name and then run the script:

```
:setvar CortexDBUSer "domain\CTX_App_User_Service_Account"
```


4 Configuration Store Parameters

Assuming that the Cortex-ConfigStore Database and relevant flows have been deployed as in the CTX-Configuration-Store Deployment Guide, the following config items should be added to the store using the Cortex-ConfigStore-Management-UI flow under an Area called “State Engine”



- FlowApiUsername - The default username for the flow API. By default this can be found in C:\Program Files (x86)\Cortex\Cortex Flow Interface Service\Innovise.Cortex.Web.Owin.dll.config under the “Allowed User” setting
- FlowApiPassword - The default password for the flow API. By default this can be found in C:\Program Files (x86)\Cortex\Cortex Flow Interface Service\Innovise.Cortex.Web.Owin.dll.config under the “Allowed Password” setting
- FlowApiUrlList - A comma separated list of possible flow API urls, the State Engine flows will use the one from the list which contains the hostname of the application server they are running on.
- MailFromAddress - The email address from which to send email notifications.
- MailServer - Fully qualified machine name of the SMTP server from which to send email notifications.
- Max-Days-ATM-Execution-Stage-Monitoring – Max Number of days a process stage is expected to run.

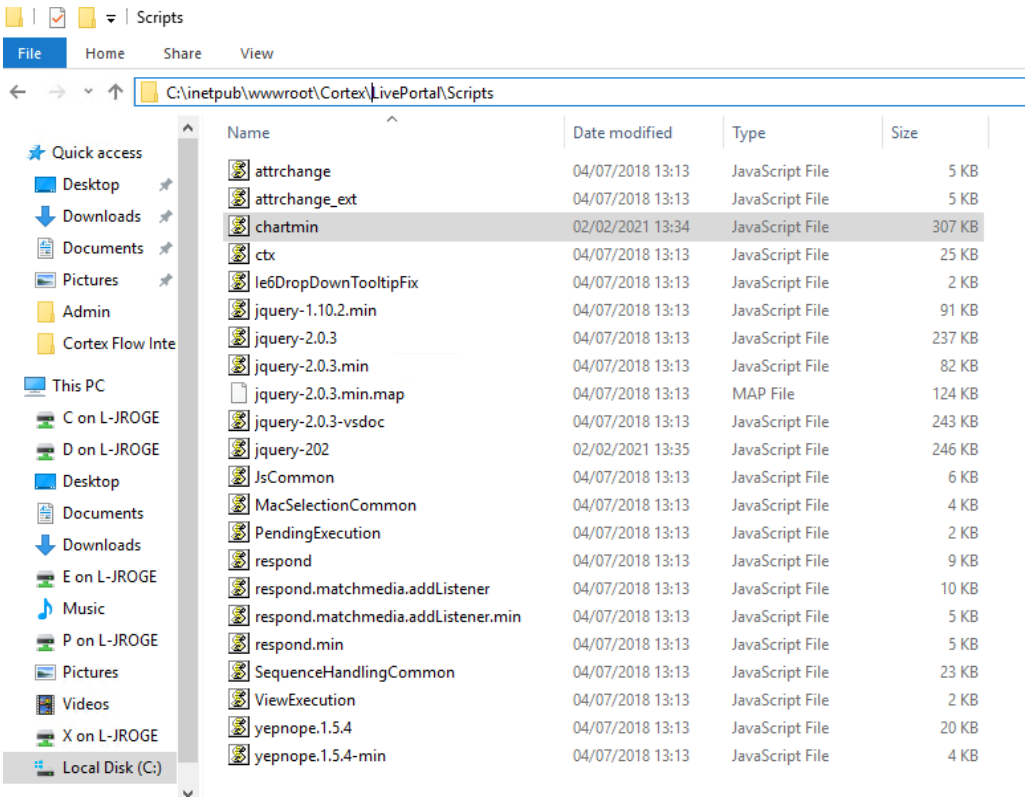
- Refer to 'CTX_ConfigStore-Data-Setup_for_Cortex-State-Engine.sql' for the sample data that could be inserted into Cortex-ConfigStore DB.

5 ChartJS JavaScript Library

For the flow State-Engine-Manage-Executions-UI to have its LivePortal UI function correctly, the ChartJS JavaScript library must be placed in the correct place on each Cortex application server running the flow.

1. Download chartmin.js from the CTX-State-Engine GitHub page
2. On each application server that will run the flow State-Engine-Manage-Executions-UI, place it in the LivePortal scripts directory.

   Depending on the installation process, this should be similar to "C:\inetpub\wwwroot\Cortex\LivePortal\Scripts"



6 Schedule the Monitoring Flow

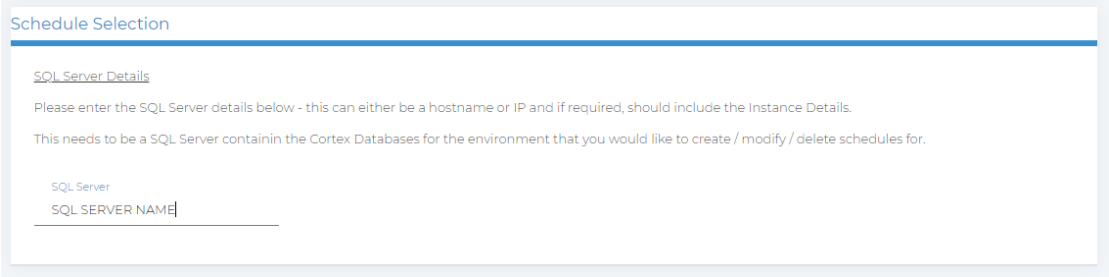
6.1 Overview

For the CTX-State-Engine module to function, the flow State-Engine-Monitor-Stage-Executions must be executed periodically. This flow evaluates the state of the flows that the state engine has called and updates the Cortex-State-Engine database with any changes since its last execution.

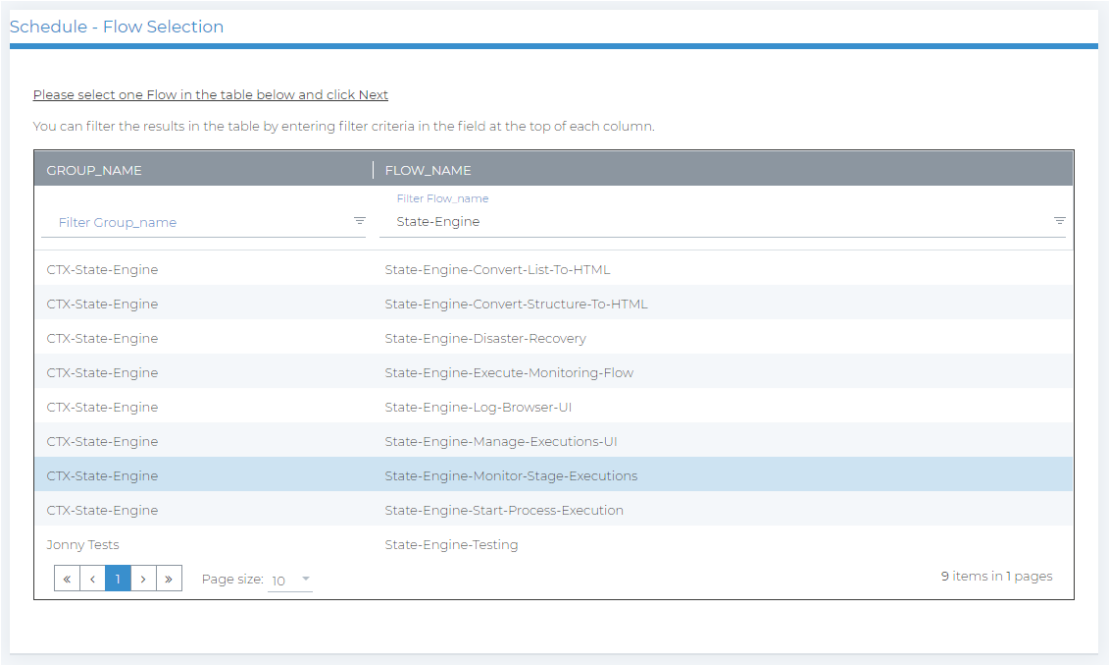
6.2 Use the Task Scheduler

To schedule the monitoring flow, take the following steps:

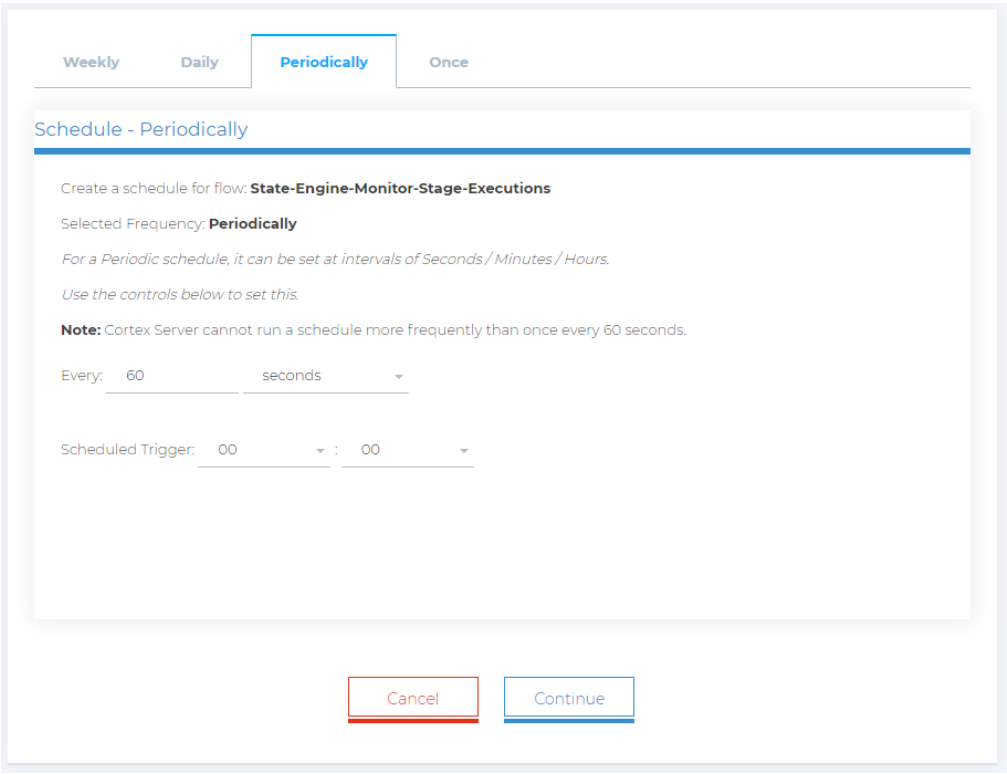
1. Ensure that the flow State-Engine-Monitor-Stage-Executions is published. This is the version that will be executed by the scheduler.
2. From LivePortal, launch the flow CTS-Manage-Schedule, the user will be presented with the following screen:



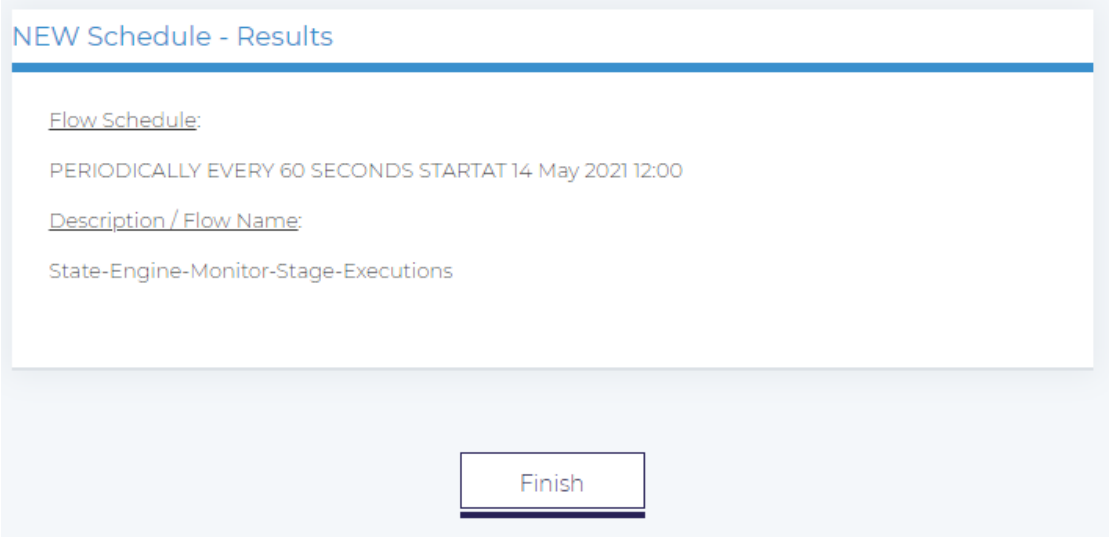
3. Enter the name of the SQL server on the same site as the Cortex application server on which we wish to execute the state engine flows. Select OK.
4. The user will be presented with a list of existing schedules if any exist. Select “Create New Schedule” at the bottom of the panel.
5. The user will be presented with a list of flows that exist on the Cortex application server on this site. Find “State-Engine-Monitor-Stage-Executions” under the Group_Name of “CTX-State-Engine”. Select Next.



6. The user will be presented with a screen allowing them to select how often the flow should be executed. The following is recommended but if a different schedule is required then that should be used:
   a. Navigate to the “Periodically” tab.
   b. Configure to execute every 60 seconds, with a Scheduled trigger being a time that the scheduled executions should start each day. Select “Continue”



7. The user will be presented with a screen allowing them to configure when this scheduled series of executions should start. Configure a date and time that is suitable. Select “Continue”
8. The user will be presented with a screen allowing them to specify details about the flow execution, including any parameters to pass to it. None are required for this flow, so if necessary, configure the initiator (i.e. who is responsible for the schedule) and select “Next”.
9. The Schedule will be configured in the Cortex Reactor database, and the user will be presented with a confirmation screen showing what was configured.



10. Select “Finish” to exit.

6.2.1 Alternative Manual Scheduling

Instead of the recommended method using the LivePortal flow above, the following SQL Script may be used, with the highlighted schedule edited, as necessary.

```
USE Reactor

--This script will schedule your flow to run on the desired schedule and
will return a task ID/
--If the script fails to complete the schedule entry, it will rollback any
changes made and report any errors.
--Please see the knowledge article on scheduling flows for the schedule
syntax and other useful information

DECLARE
--1. Task Schedule - Specify when you want your flow to run
@TSK_SCHED AS VARCHAR(4000) = 'PERIODICALLY EVERY 60 Seconds STARTAT 14 May
2021 12:00'

--2. Flow Name - Provide your flow name
, @TSK_FLW_NME AS VARCHAR(MAX) = 'State-Engine-Monitor-Stage-Executions'

--3. Flow Description - Provide a description of your flow schedule
, @TSK_DESC AS VARCHAR(256) = 'State Engine Monitoring flow schedule'

--4. Flow Parameters - Provide any starting parameters for your flow in a
structure format. e.g. structure (G_NAME:"Josh", G_AGE:34)
, @TSK_PARAMS AS VARCHAR(MAX) = ''

--Do not make changes to any of the below code. This may have unintended
consequences
, @TSK_EXE_ID AS INT
, @TSK_PARAM_FLOW_NAME_ID AS INT = (SELECT ID FROM CFG_PARAMETER_TYPE WHERE
PARAMETER_NAME = 'AUTOMATOR-RULE-NAME')
, @TSK_PARAM_PARAMS_ID AS INT = (SELECT ID FROM CFG_PARAMETER_TYPE WHERE
PARAMETER_NAME = 'AUTOMATOR-RULE-PARAMETERS')
,   @TSK_TYPE_ID AS INT = (SELECT ID FROM CFG_TASK WHERE TASK_TYPE =
'AUTOMATOR-RULE-EXECUTION')
BEGIN TRANSACTION;
BEGIN TRY

--Add entries to CFG_TASK_EXECUTION, then selects ID
       BEGIN
              INSERT   INTO   [Reactor].[dbo].[CFG_TASK_EXECUTION](TSK_ID,
DESCRIPTION)
              VALUES (@TSK_TYPE_ID, @TSK_DESC);
              SELECT @TSK_EXE_ID = SCOPE_IDENTITY();
       END
      SELECT @TSK_EXE_ID AS [Task Exe ID];

--Add entries to CFG_TASK_PARAMETER


       BEGIN
```



```
              INSERT  INTO  [Reactor].[dbo].[CFG_TASK_PARAMETERS]  (TEX_ID,
PRM_ID, PARAMETER_VALUE)
              VALUES (@TSK_EXE_ID, @TSK_PARAM_FLOW_NAME_ID, @TSK_FLW_NME);
              INSERT  INTO  [Reactor].[dbo].[CFG_TASK_PARAMETERS]  (TEX_ID,
PRM_ID, PARAMETER_VALUE)
              VALUES (@TSK_EXE_ID, @TSK_PARAM_PARAMS_ID, @TSK_PARAMS);
       END

--Add entry to CFG_TASK_SCHEDULE
       BEGIN
              INSERT  INTO  [Reactor].[dbo].[CFG_TASK_SCHEDULE]  (TEX_ID,
SCHEDULE)
              VALUES (@TSK_EXE_ID, @TSK_SCHED);
       END
END TRY

BEGIN CATCH
       PRINT ERROR_MESSAGE()
       SELECT ERROR_MESSAGE();

       IF @@TRANCOUNT > 0
              ROLLBACK TRANSACTION;
END CATCH;
IF @@TRANCOUNT > 0
       COMMIT TRANSACTION;
```

