Rilevare se il License Server è in Grace Period

Introduzione

Questa TN descrive le operazioni da seguire per rilevare da System Platform se il License Server è in Grace Period e la sua data di scadenza, tramite script da Application Server. Vengono mostrati due metodi:

1. tramite interrogazione del database sqlite LicenseManagerDb.db in ODBC;
2. tramite richiesta https verso le Web API esposte dal License Server

Versioni

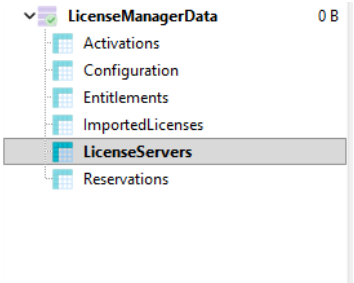
Quanto descritto in questa TN è stato testato sulla versione Application Server 2026 ma vale per le seguenti versioni:

- Application Server >= 2023 R2
- License Manager >= 4.0.1

Leggere il LicenseManagerDb.db sqlite in ODBC

Struttura del LicenseManagerDb.db

Il License Manager salva molte informazioni sulle operazioni, sulle licenze e sullo stato del license server nel file sqlite LicenseManagerDb.db , nel percorso:
C:\ProgramData\AVEVA\Licensing\License Manager\Data\
Il database ha la seguente struttura, in particolare ci interessa la tabella *LicenseServers*:



1 Server_Id	2 Name	3 HostId	4 Display_Name	5 Server_State	6 Server_Lice...	7 Alternate_Ser...	8 ServerType	9 InUse_Licenses	10 Is_Default	11 Comments	12 SyncStatus	13 GracePeriod
1	SP2026	SP2026_UODFP7T5	SP2026	0	2		0	0	0			11/25/2026 08:33:43

Da cui si può ricavare il License Server ID (colonna HostID), e lo stato del Grace Period (colonna GracePeriod). Quest ultimo risulterà vuoto se non in GracePeriod, altrimenti visualizzerà la data di scadenza o lo stato “expired” se scaduto il periodo di validità.

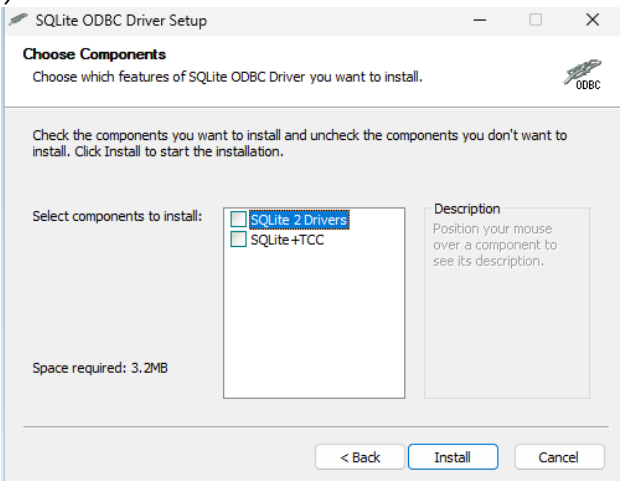
N.B. Il file è un file di sistema, è pertanto raccomandato ***non*** eseguire Query o SQL Injection che alterino l’integrità del database.

Installazione e configurazione Driver ODBC

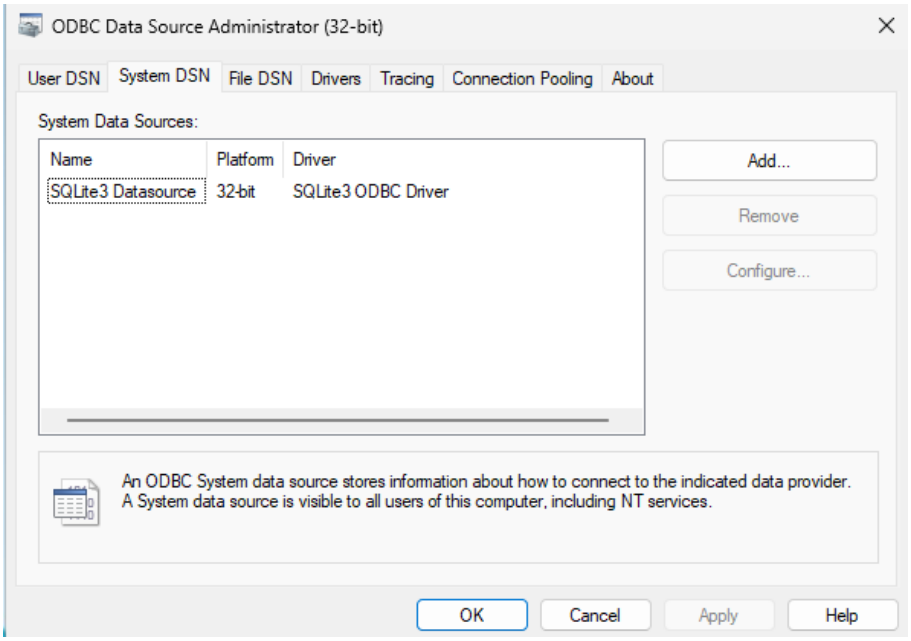
E' necessario installare un driver ODBC per interrogare database sqlite LicenceManagerDb.db in ODBC.

1. Scaricare un driver ODBC, ad esempio questo al seguente LINK (ref. Technote AVEVA https://softwaresupportsp.aveva.com/en-US/knowledge/details/000032549?lang=en_US)
2. Installare il driver sulla macchina dove è deployoato un AppEngine che eseguirà lo script, lanciando il file .exe come Amministratore;
3. Seguire il setup guidato lasciando tutto di default fino alla schermata **Install** (senza selezionare SQLite 2 Drivers o SQLite+TCC):





4. Verificare che il driver sia stato installato correttamente nelle ODBC Data Sources (32-bit), facendo una ricerca da Start di Windows "ODBC":



5. A questo punto il driver è stato installato e configurato correttamente.

Implementazione Script per Lettura del db sqlite LicenceManagerDb.db

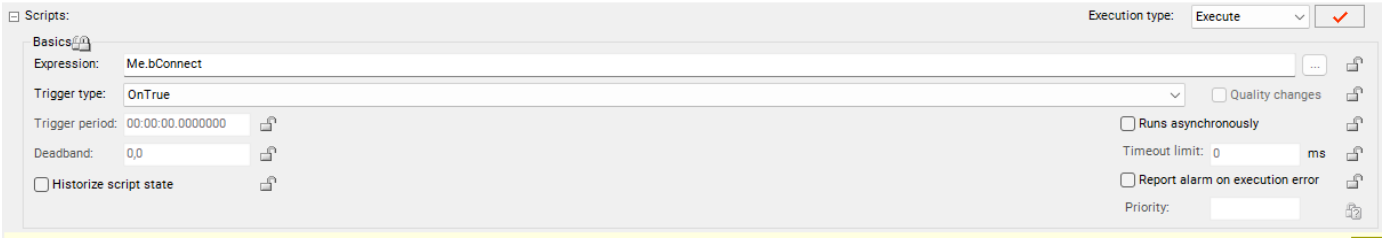
1. In questo esempio creiamo **un'istanza** User Defined chiamata *sqliteGracePeriod* all'interno della System Platform, con gli **Attributi**:
   - **bConnect** (bool per attivazione dello script);

- **bGPAlarm** (bool di Allarme se rileviamo che il licensing è in Grace Period); settiamolo come allarme e come condizione di allarme State Alarm;
- **sHostID** (stringa a cui verrà assegnato il nome del License Server ID);
- **sGPDateTime** (stringa con data ora rilevata di scadenza del Grace Period)
- **sPath** (stringa hard coded con al suo interno il path del LicenseManagerDb.db)

il file LicenseManagerDb.db si trova in:

"C:\ProgramData\AVEVA\Licensing\License Manager\Data\LicenseManagerData.db;"

2. all'interno dell'istanza creiamo lo script ***sqliteRead*** con le seguenti condizioni di avvio:
   Expression: Me.bConnect
   Trigger Type: OnTrue



3. Inseriamo quindi il seguente script:

```
dim connectionString as string;
dim conn as System.Data.Odbc.OdbcConnection;
dim reader as System.Data.Odbc.OdbcDataReader;
dim cmd as System.Data.Odbc.OdbcCommand;
dim sql as string;
dim res as string;

Me.bConnect = false;

connectionString = "DRIVER=SQLite3 ODBC
Driver;Database="+Me.sPath+";LongNames=0;Timeout=1000;NoTXN=0;SyncPragma=NORMAL;
StepAPI=0;";

LogMessage("ConnString = " + connectionString);

try
    conn = new System.Data.Odbc.OdbcConnection(connectionString);
    conn.Open();

    sql = "SELECT HostId, GracePeriod FROM LicenseServers;";
    cmd = System.Data.Odbc.OdbcCommand(sql, conn);
    reader = cmd.ExecuteReader();

      LogMessage("is Ok until here? = " + sql + " - " + connectionString);


    while (reader.Read())

        Me.sHostID = reader.GetString(0);
            Me.sGPDateTime = reader.GetString(1);
            LogMessage("HostID = " + Me.sHostID + ", GPDateTime = " +
Me.sGPDateTime);

      endwhile;

      if Me.sGPDateTime <> "" then
```

```
            Me.bGPAlarm.Description = Me.bGPAlarm.Description + " " +
Me.sGPDateTime;
            Me.bGPAlarm = true;
            LogMessage(Me.bGPAlarm.Description);
      endif;

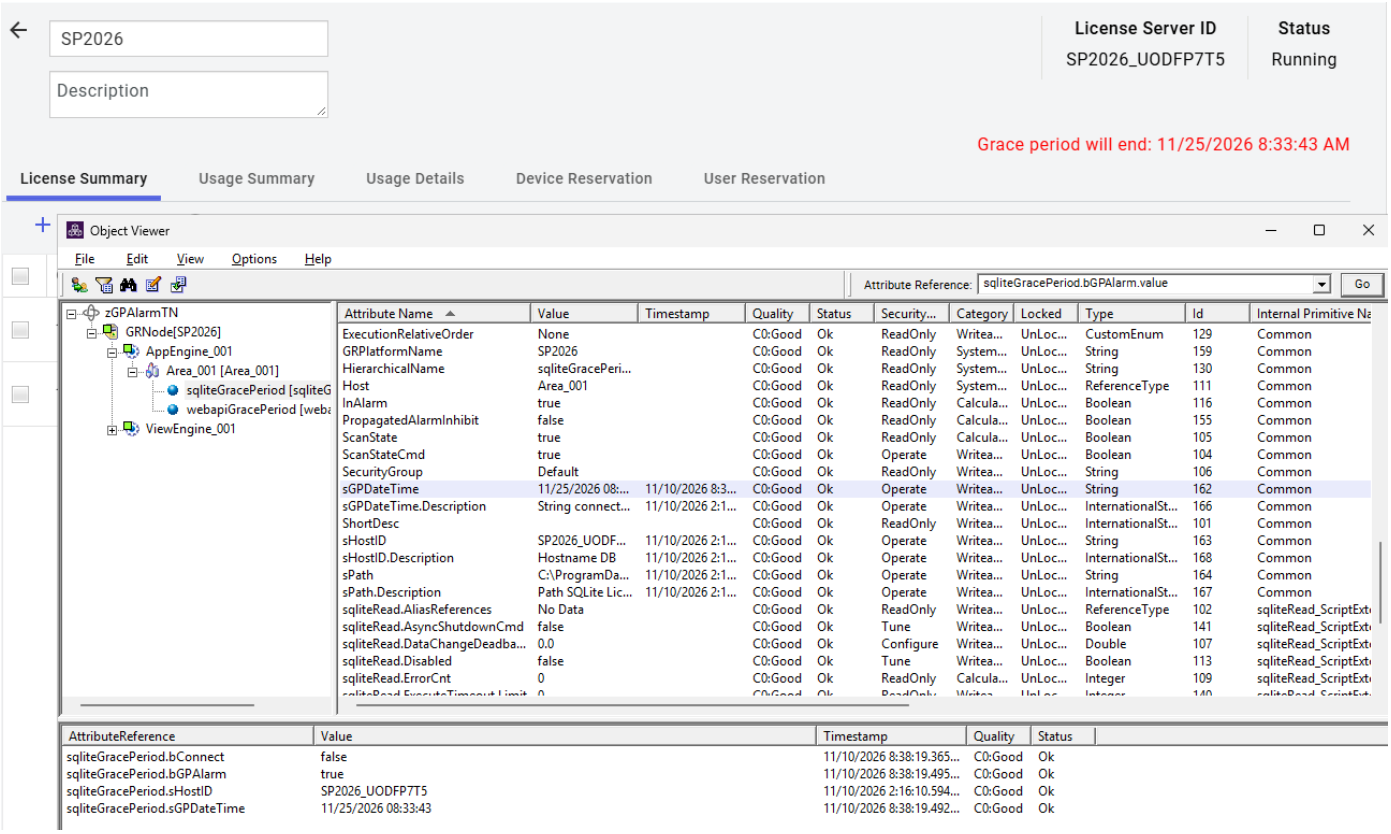
catch
    LogError(error);
endtry;

if reader <> null and not reader.IsClosed then
      reader.Close();
endif;

if cmd.Connection.State == System.Data.ConnectionState.Open then
      cmd.Connection.Close();
endif;
```

N.B. Lo script di cui sopra va preso “as is” e costituisce un esempio da adattare alle caratteristiche specifiche del proprio ambiente e della propria applicazione, integrandolo con logiche che non vadano ad impattare il corretto funzionamento del sistema.

Attivando il booleano bConnect da ObjectViewer, possiamo constatare che vengono valorizzati lo stato del grace period e la data ora di scadenza in conformità a quanto visualizzato sul License Manager:



Implementazione Script per chiamata API del License Server

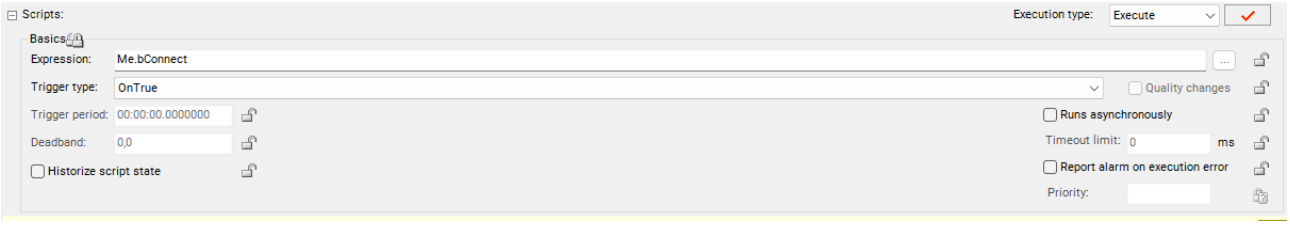
E' possibile interrogare lo stato del Grace Period attraverso le Web API messe a disposizione del License Server, all'indirizzo: https://<hostname>:55559/api/1.0/binding/
Se il License Server è andato in grace period, viene visualizzato il seguente json di risposta:

```
{
  "bindingPolicy" : "1296000 seconds",
  "bindingStatus" : "soft break",
  "gracePeriodExpiry" : "2026-11-25T16:33:43Z"
}
```

In particolare, la voce "bindigStatus" passa da "ok" a "soft break" e viene restituito in aggiunta la voce "gracePeriodExpiry" : "2026-11-25T16:33:43Z", con la data di scadenza del grace period in formato ISO.
Andiamo quindi a fare la richiesta da uno script di un'istanza User Defined da Application Server.

1. Creiamo un'istanza User Defined chiamata *webapiGracePeriod*
2. Al suo interno creiamo gli attributi:
   - **bAPI_GPAlarm:** Bool di allarme se rileviamo il Licensing in Grace Period, con State Alarm attivo;
   - **bConnect:** Bool di attivazione dello script;
   - **sAPI_GPExpiryDate:** String con data ora di scadenza del Grace Period

3. Creiamo lo script *apiGracePeriod* che come Expression abbia Me.bConnect e Trigger Type OnTrue:



4. Inseriamo il seguente script:

```
dim req as System.Net.HttpWebRequest;
dim resp as System.Net.HttpWebResponse;
dim reader as System.IO.StreamReader;
dim json as string;
dim rawDate as string;
dim idx as integer;

' Resetta il trigger OnTrue
Me.bConnect = false;

try
    req =
System.Net.HttpWebRequest.Create("https://localhost:55559/api/1.0/binding/");
    req.Timeout = 5000;
    resp = req.GetResponse();

    if resp.StatusCode == System.Net.HttpStatusCode.OK then
```

```
        reader = new System.IO.StreamReader(resp.GetResponseStream());
        json = reader.ReadToEnd();
        reader.Close();

        ' 1. Prima verifichiamo se è in soft break (Grace Period)
        if json.Contains("soft break") then
            Me.bAPI_GPAlarm = true;

            idx = json.IndexOf("gracePeriodExpiry");
            if idx > 0 then
                idx = json.IndexOf("""", json.IndexOf(":", idx)) + 1;
                rawDate = json.Substring(idx, json.IndexOf("""", idx) - idx);

                Me.sAPI_GPExpiryDate =
System.Convert.ToDateTime(rawDate).ToString("dd/MM/yyyy HH:mm:ss");
                LogMessage("License Server in soft break - Scadenza: " +
Me.sAPI_GPExpiryDate);
            endif;

        ' 2. Altrimenti (qualsiasi altro stato, es. ok) consideriamo tutto
regolare
        else
            Me.bAPI_GPAlarm = false;
            Me.sAPI_GPExpiryDate = "-";
            LogMessage("License Server OK.");
        endif;
    endif;

catch
    LogError(error);
endtry;

' Chiusura risorse
if resp <> null then
    resp.Close();
endif;
```

N.B. Lo script di cui sopra va preso “as is” e costituisce un esempio da adattare alle caratteristiche specifiche del proprio ambiente e della propria applicazione, integrandolo con logiche che non vadano ad impattare il corretto funzionamento del sistema.

E’ Inoltre necessario manutenere e tenere aggiornato lo script a fronte di upgrade del software, poiché le folder e le API presenti nel seguente documento potrebbero essere soggetti a cambiamenti nelle future release.

Referenze

- *AVEVA™ Enterprise Licensing Guide*
- https://softwaresupportsp.aveva.com/en-US/knowledge/details/000032549?lang=en_US