Export Lista Allarmi Storici da SQL Server in file .csv

Introduzione

Questa TN mostra come esportare la lista allarmi tramite script da un simbolo ArchestrA da client InTouch, accedendo agli allarmi storicizzati in Historian tramite SQL Server, puntando al Database Runtime.
E' possibile esportare la lista allarmi con configurazione Alarm DB logger puntando al Database WWALDB

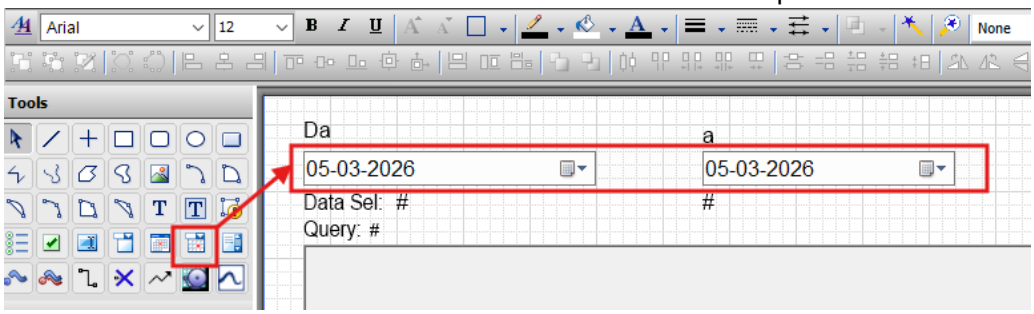
Versioni

Quanto descritto in questa TN è stato verificato sulle seguenti versioni:

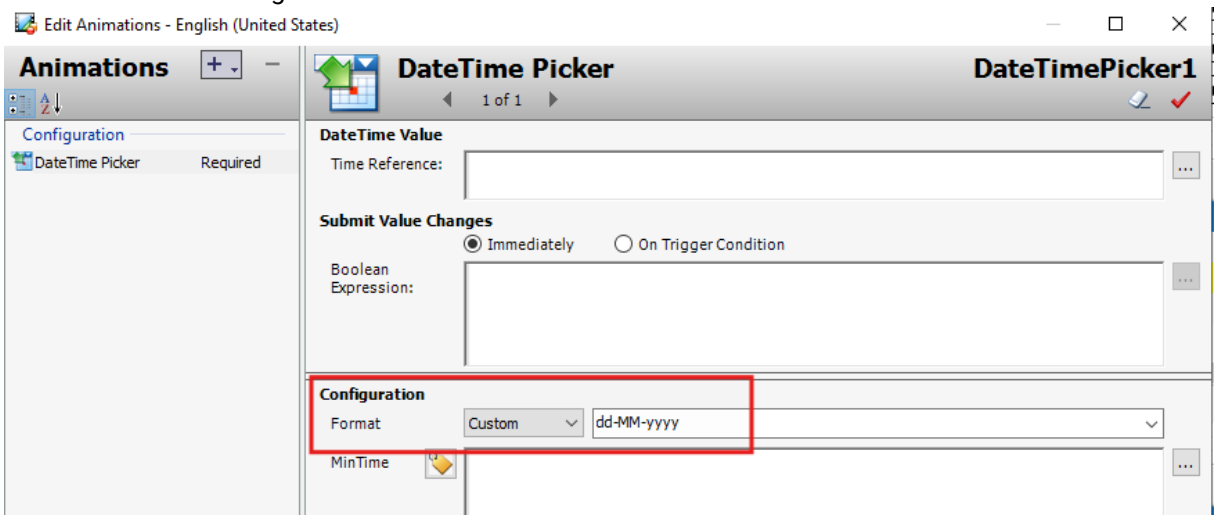
- Dalla versione 2017 U3 SP1 in avanti

Impostazione Simbolo Grafico Archestra

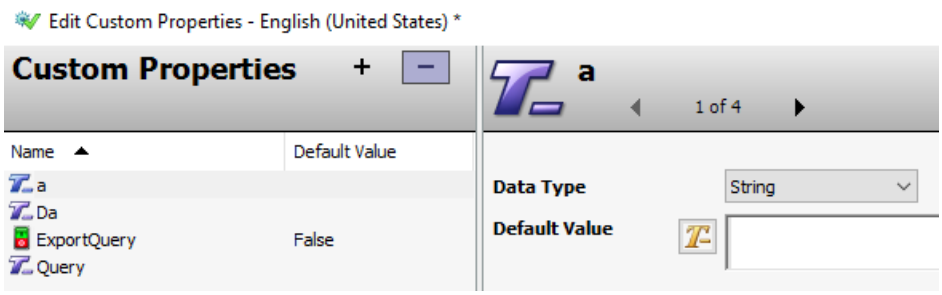
1. Embeddare in un simbolo ArchestrA dei controlli Datetime Picker per selezione data inizio-fine:



2. Formattarli come di seguito:



3. Creare 4 custom property, 3 stringa chiamate “Da”, “a”, “Query”, e una Booleana chiamata “ExportQuery”:

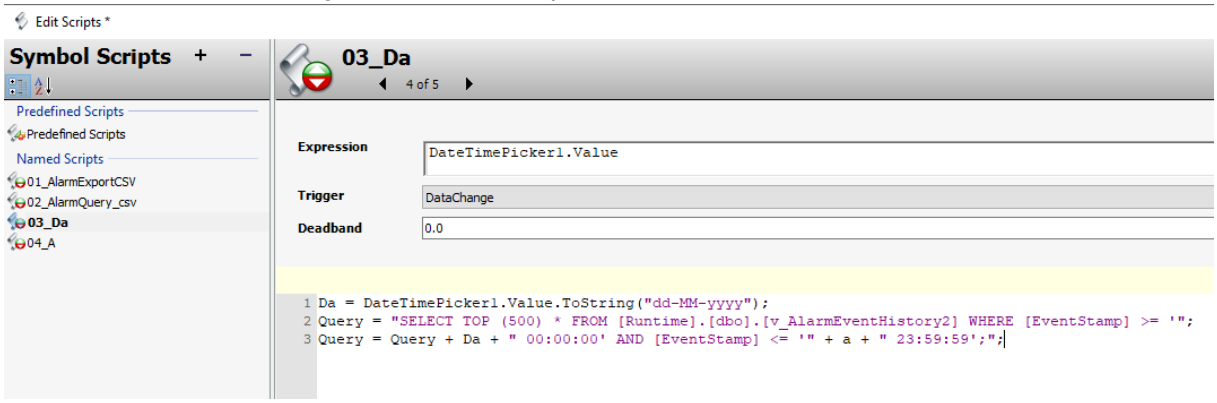


4. Sempre all’interno del simbolo, premere F10 e inserire due Named Script chiamati “03_Da” e “04_a”, che come condizione abbiano DateTimePicker1.Value e DateTimePicker2.Value (i nomi dei DateTimePicker), e si azionino al DataChange e contengano il seguente script:

```
Da = DateTimePicker1.Value.ToString("dd-MM-yyyy");
Query = "SELECT TOP (500) * FROM [Runtime].[dbo].[v_AlarmEventHistory2]
WHERE [EventStamp] >= '";
Query = Query + Da + " 00:00:00' AND [EventStamp] <= '" + a + "
23:59:59';";
```

In modo che venga aggiornata la data di inizio e fine e la relativa Query, in questo modo:

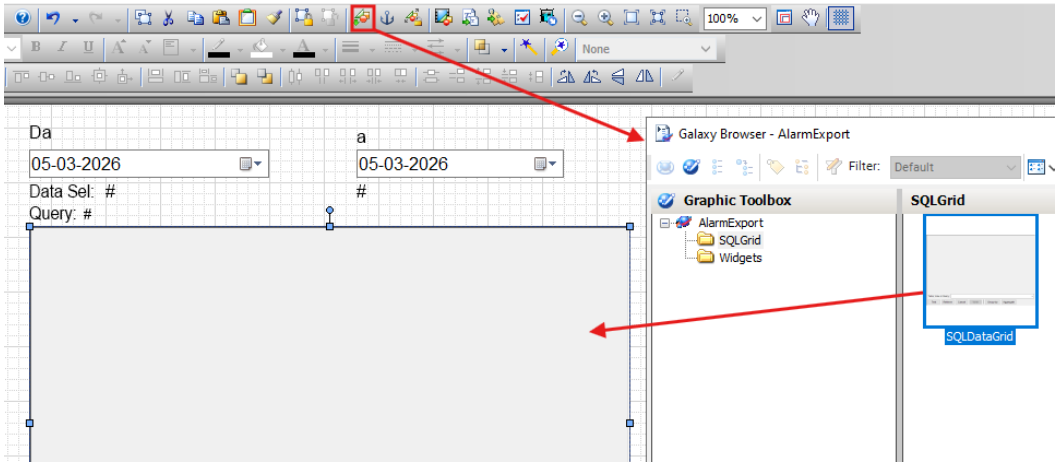
1. Si punta alla vista del database Runtime di Historian dove vengono storicizzati gli allarmi;
2. Viene selezionato il giorno intero dalla mezzanotte alle 23:59:59 dello stesso giorno in caso di selezione dello stesso giorno dei datetimepicker;



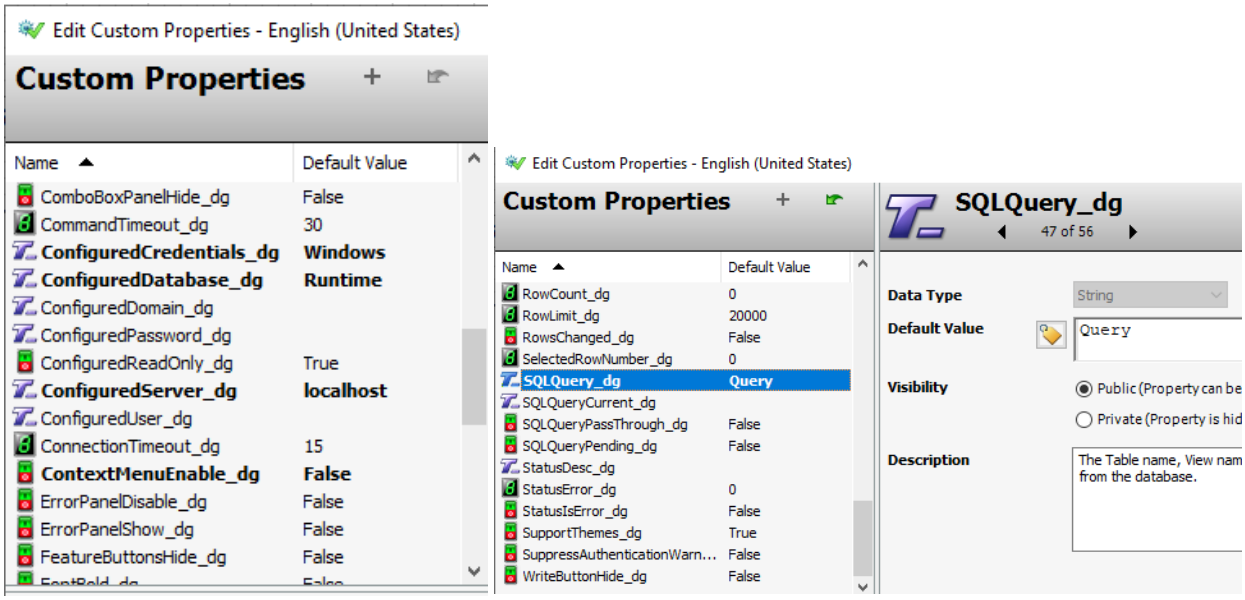
In caso di necessità, cambiando la formattazione dei DateTimePicker aggiungendo hh:mm:ss, è possibile inserire anche l’ora e i minuti di data-ora inizio e fine, formattando la data-ora anche nello script in:
`DateTimePicker1.Value.ToString("dd-MM-yyyy hh:mm:ss");`

E’ inoltre possibile puntare alla tabella degli allarmi storici del Database WWALMDB se si utilizza la storicizzazione degli allarmi tramite Alarm DB Logger, puntando alla tabella v_AlarmHistory, cambiando la query in:
`“SELECT TOP (500) * FROM [WWALMDB].[dbo].[v_AlarmHistory]” […]`

5. Ora Embeddiamo un SQLDataGrid per visualizzare a schermo gli allarmi risultanti dalla Query:



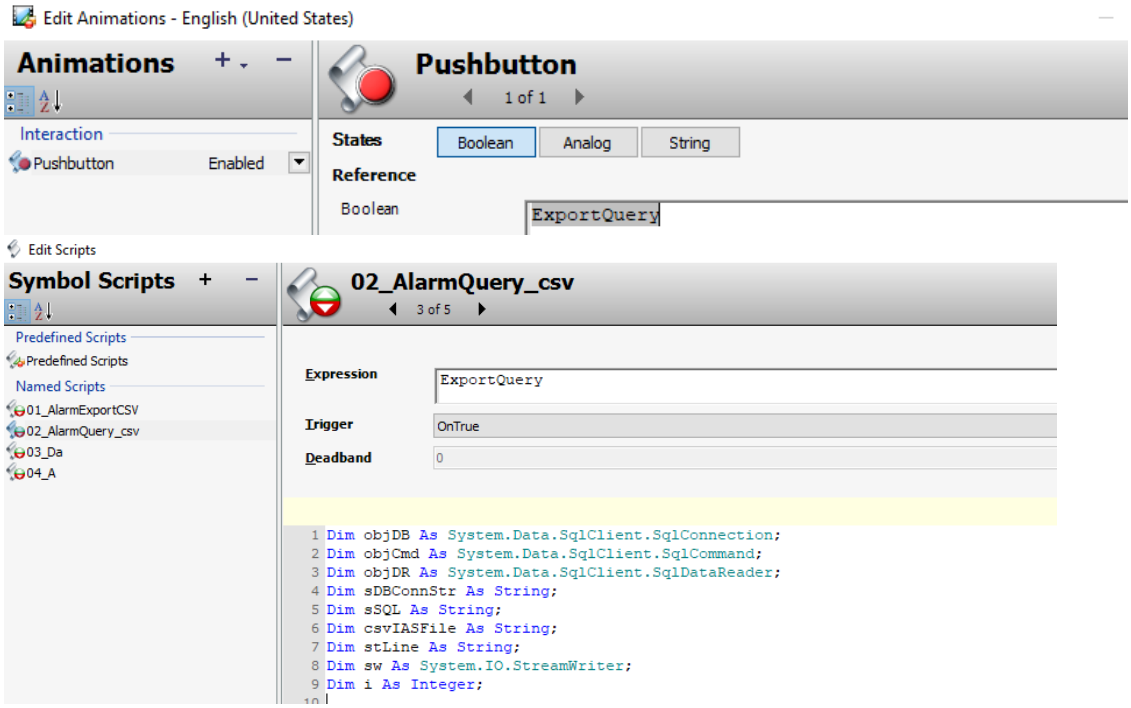
6. Nel mio caso, ho configurato il database Historian in locale e ho optato per l'autenticazione Windows Integrated; vanno quindi compilate le custom property dell'SQLDataGrid in questo modo:



Importante: la Query ricavata dalla custom property tramite script, va passata come default value dinamico alla custom property SQLQuery_dg del controllo SQLDataGrid.

7. Inseriamo ora due Pulsanti,
    a. uno denominato "Retrieve" per eseguire la query e popolare la griglia, con l'animazione Action Script e lo script seguente:
    ```
    SQLDataGrid1.CmdRetrieve_dg = true;
    ```
    b. uno denominato "Export", per ri-eseguire la stessa query e effettivamente creare l'export degli allarmi creando un file csv.

Per il secondo, attiveremo lo script settando a True la custom property “ExportQuery” creata all’inizio, che sarà la condizione di partenza di un Named Script chiamato “01_AlarmQuery_csv”:



Inseriamo lo script seguente:

```
Dim objDB As System.Data.SqlClient.SqlConnection;
Dim objCmd As System.Data.SqlClient.SqlCommand;
Dim objDR As System.Data.SqlClient.SqlDataReader;
Dim sDBConnStr As String;
Dim sSQL As String;
Dim csvIASFile As String;
Dim stLine As String;
Dim sw As System.IO.StreamWriter;
Dim i As Integer;

ExportQuery = false;

sDBConnStr = "Data Source=localhost;Initial Catalog=Runtime;Integrated
Security=True";

sSQL = Query;

csvIASFile = "C:\AlarmExport\AlarmExport_" +
System.DateTime.Now.ToString("dd-MM-yyyy_HH-mm-ss") + ".csv";

Try
    '' Connessione e apertura del database
    objDB = New System.Data.SqlClient.SqlConnection(sDBConnStr);
    objDB.Open();

    '' Invoke del commando SQL
    objCmd = New System.Data.SqlClient.SqlCommand(sSQL, objDB);
    objDR = objCmd.ExecuteReader();

    If objDR.HasRows Then
        sw = System.IO.File.CreateText(csvIASFile);
```

```
        '' --- GESTIONE DINAMICA INTESTAZIONI ---
        stLine = "";
        For i = 0 To objDR.FieldCount - 1
            stLine = stLine + objDR.GetName(i);
            If i < (objDR.FieldCount - 1) Then stLine = stLine + ","; EndIf;
        Next;
        sw.WriteLine(stLine);

        '' --- LETTURA DATI ---
        While objDR.Read()
            stLine = "";
            For i = 0 To objDR.FieldCount - 1
                '' Aggiunta gestione per i valori Null per evitare crash
                If objDR.IsDBNull(i) Then
                    stLine = stLine + "NULL";
                Else
                    stLine = stLine + objDR.GetValue(i).ToString();
                EndIf;

                If i < (objDR.FieldCount - 1) Then stLine = stLine + ",";
EndIf;
            Next;
            sw.WriteLine(stLine);
        EndWhile;

        sw.Close();
        sw.Dispose();
    EndIf;

    objDR.Close();
    objDB.Close();

Catch
    '' Log dell'errore
    LogMessage(error);

    '' Chiusura degli oggetti in caso di errore
    sw.Close();
      sw.Dispose();
    If objDB.State == System.Data.ConnectionState.Open Then objDB.Close();
EndIf;
EndTry;
```

Lo script:

1. costruisce la stringa di connessione (nel mio caso in Windows Autentication); nel caso di password diversa non metterla in chiaro nello script ma costruire la stringa altrove (ad esempio in degli attributi se Intouch per System Platform, oppure in delle tag intouch);
2. gestisce eventuali errori di connessione al database tramite Try Catch;
3. Aggiunge a un file le intestazioni delle colonne della tabella del db, tramite il metodo `System.IO.File.CreateText();`
4. Mentre legge, aggiunge le righe al file;
5. Alla termine della scrittura del file, chiude le connessioni al database.

In questo modo verrà creato un file csv con anche data-ora di esecuzione dell'export.

N.B.: La query presa in esame fa una select su un massimo di 500 righe per evitare un blocco del processo della grafica (in questo caso lo script viene eseguito lato client che ha un solo processore).

Nel caso serva esportare più allarmi, si può demandare lo script all'application server se si utilizza Intouch per System Platform, oppure delegando a uno script while true la costruzione del file csv.

Referenze

AVEVA™ The SQLDataGrid graphic: https://docs.aveva.com/bundle/sp-appserver/page/385863.html

AVEVA™ Historian Retrival Guide: https://docs.aveva.com/bundle/sp-historian/page/1393205.html

AVEVA™ Scripting Reference: https://docs.aveva.com/bundle/sp-appserver/page/818109.html