Citect Alarm Browser for ScadaPhone

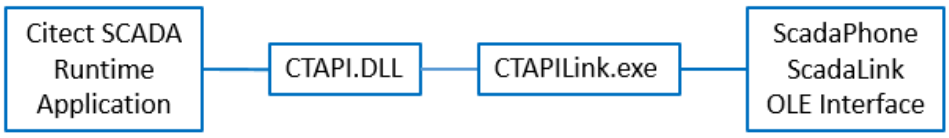
Overview

To facilitate project configuration, ScadaPhone implements a **Citect Alarm Browser, CTAPILink**. This browser can reduce ScadaPhone project configuration from hours down to minutes when ScadaPhone is to be used in conjunction with **Citect SCADA**. The Citect Alarm Browser uses **Citect's CTAPI interface (CiTect Application Program Interface)** to obtain a list of all alarms configured in the Citect project and then facilitate the selection and configuration of alarms to be serviced by ScadaPhone.

In most cases, ScadaPhone communicates with SCADA servers via **OPC/OLE**; however, **Citect** has another, more customized, interface for client applications. Citect's custom interface is called **CTAPI** (**CiTect Application Program Interface**).

In order to interface with **CTAPI**, client applications must load the **CTAPI Dynamic Link Library** (**CTAPI.DLL**), establish a connection, and then request data via calls to functions defined within **CTAPI.DLL**.

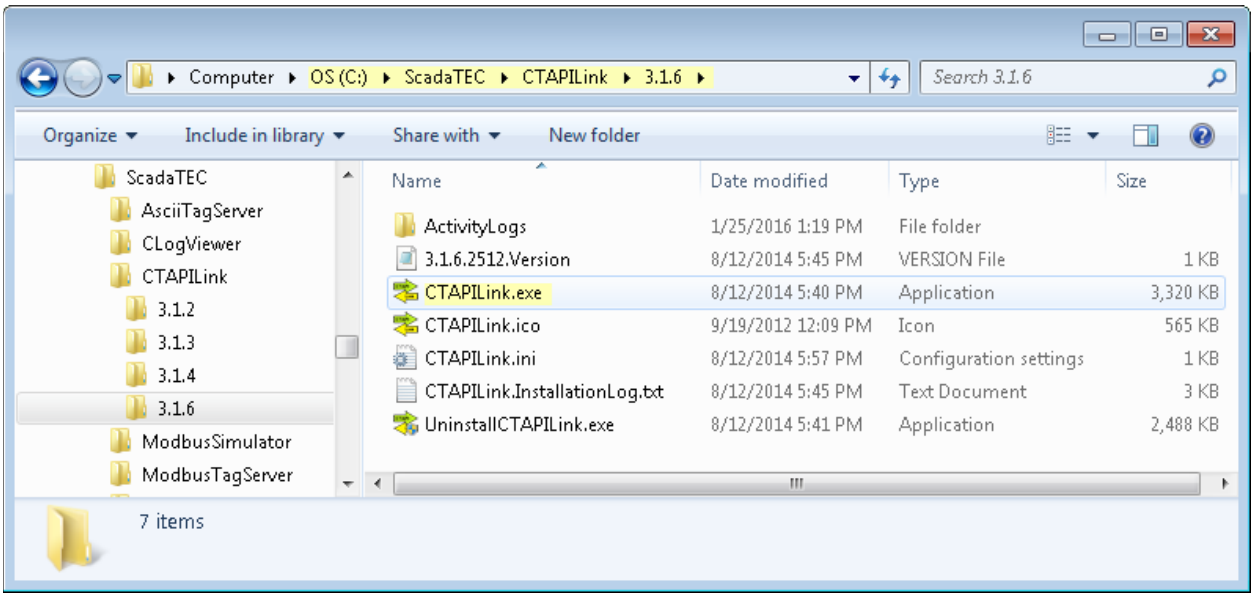
The **DLL** interface is convenient, but it doesn't mesh well with ScadaPhone's **ScadaLink interface** which is built primarily on **OLE Automation**. To bridge this communications gap without adding undesired complexity to ScadaPhone's ScadaLink interface, the **CTAPILink** auxiliary application was created; CTAPILink loads the CTAPI.DLL to interact with Citect SCADA Runtime and simultaneously provides an OLE server interface for ScadaPhone's ScadaLink interface:



The **CTAPILink.exe** is an auxiliary application which should be installed separately from ScadaPhone.

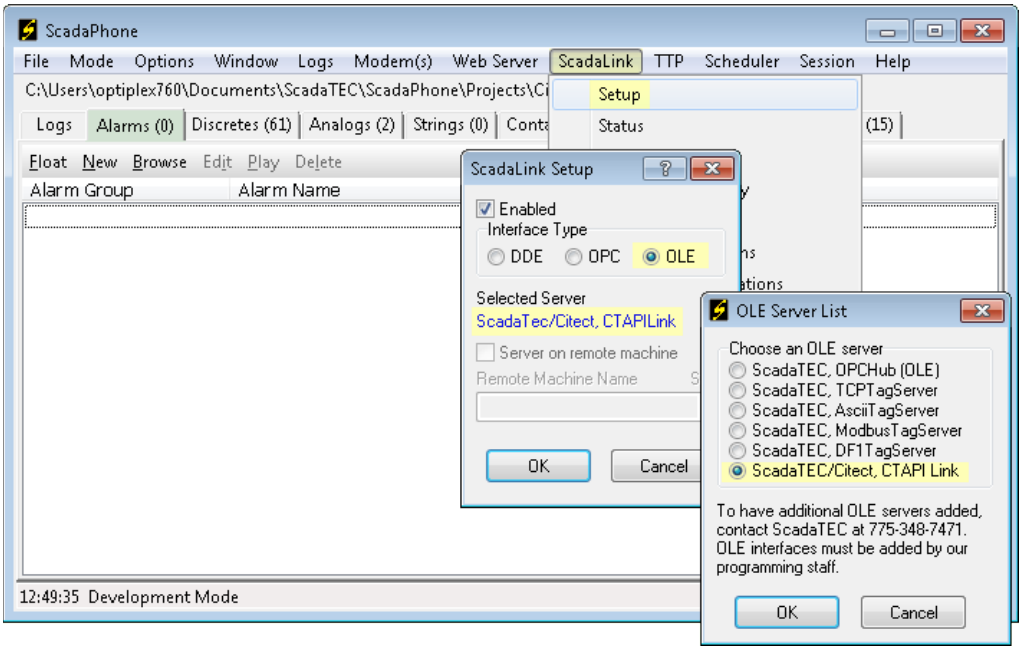
CTAPILink usually does not need to be launched by the user or by any script; it is launched automatically whenever ScadaPhone needs to use it. The only exception is during initial system configuration, **CTAPILink** may need to be configured in order to locate the **CTAPI.DLL** file in **Citect's directory**

structure. To make sure that CTAPILink is properly configured to locate CTAPI.DLL, launch CTAPILink from the **Windows File Explorer**.



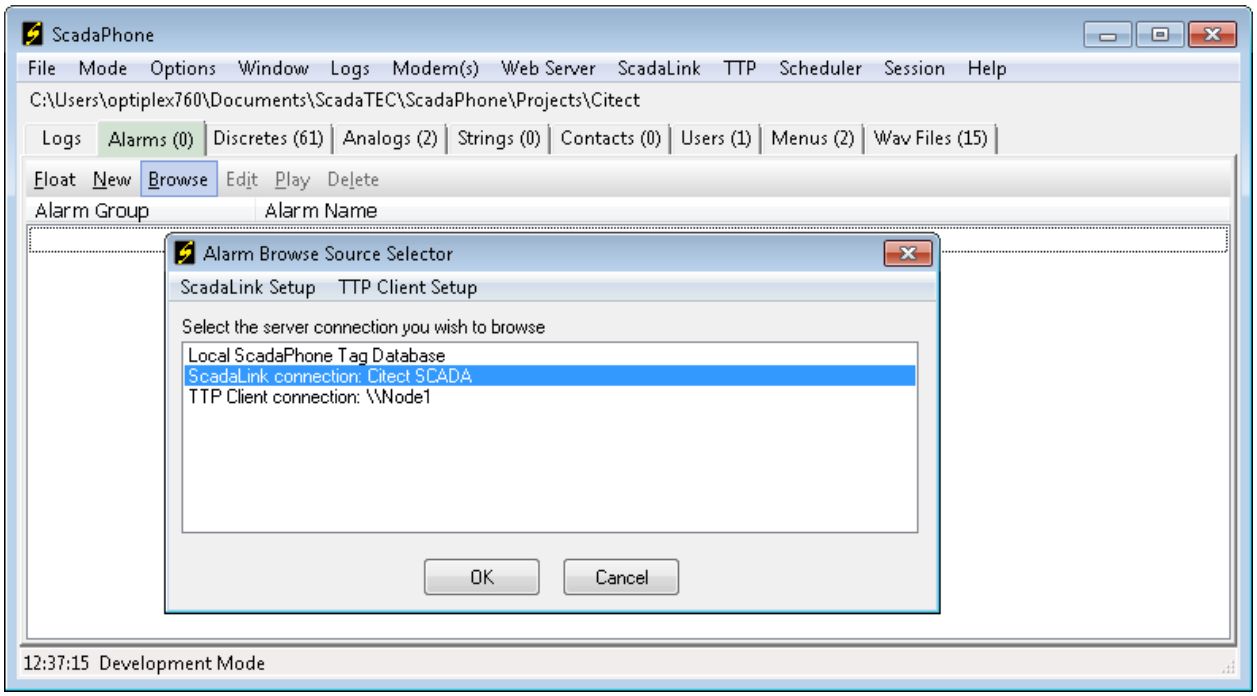
Configure ScadaLink Interface:

To access the **Citect Alarm Browser**, you must first configure ScadaPhone's **ScadaLink interface** to communicate with **Citect** via **CTAPILink**. To do this, launch ScadaPhone, create a new project, and configure the **ScadaLink Setup** as follows:



Selecting CTAPILink from the Main Menu

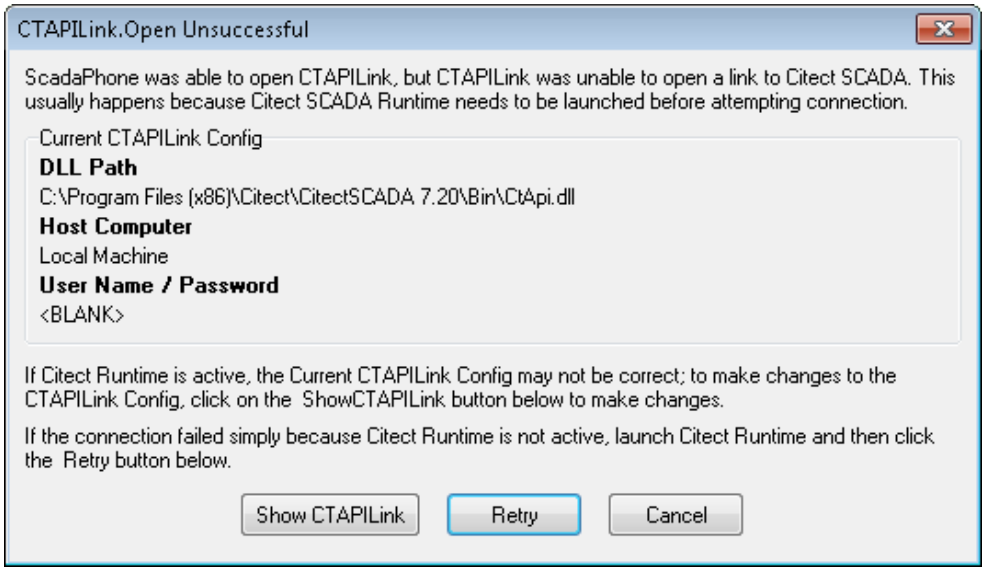
Next, select the **Alarms** tab on ScadaPhone's Main Window and click the **Browse** menu item to open the **Alarm Browse Source Selector** window:



Alarm Browser Source Selector

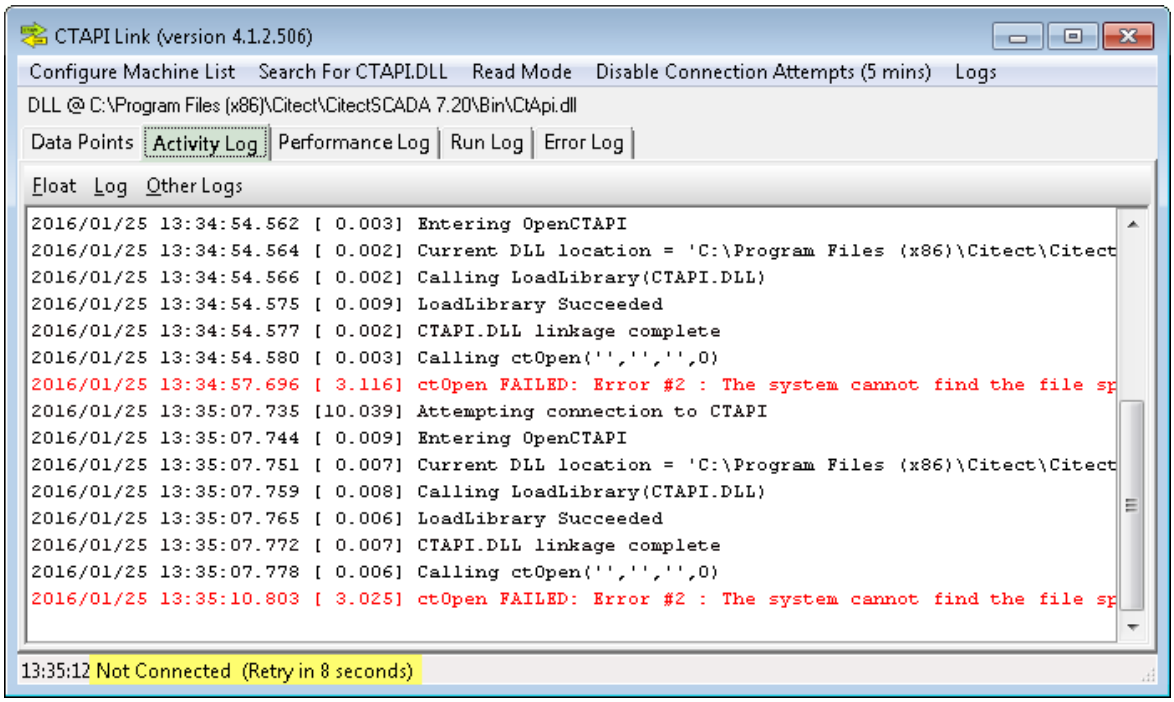
Choose the **ScadaLink connection: Citect SCADA** option from the **Alarm Browse Source Selector**.

If **Citect SCADA Runtime** is not active, you will see the following window:



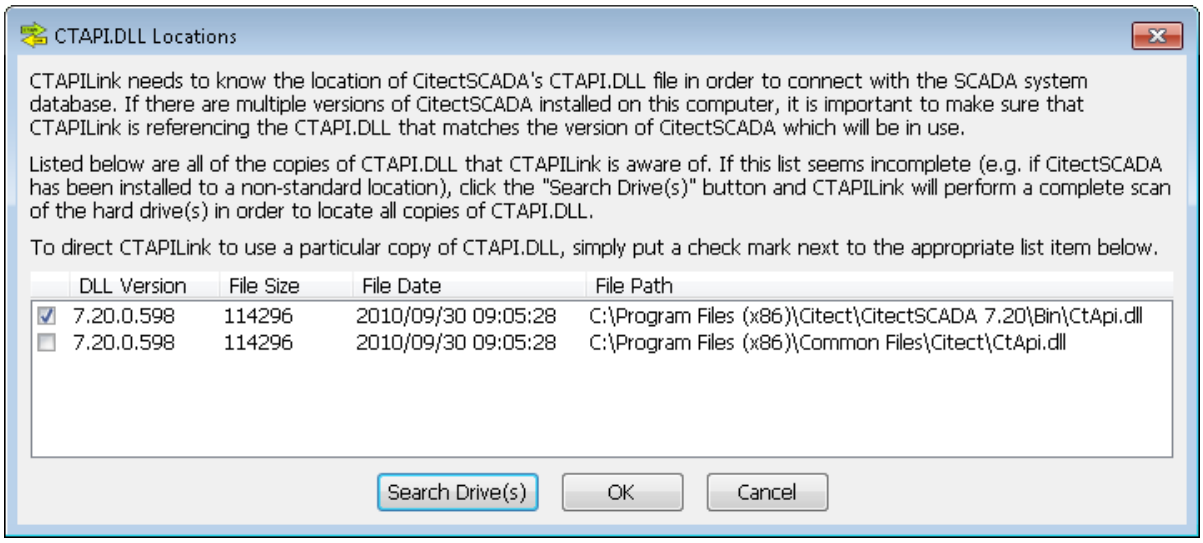
CTAPILink Unsuccessful Message if Citect is not in Runtime

If **Citect SCADA Runtime** is *not active* when **CTAPILink** is launched, the status bar at the bottom of **CTAPILink's** main window will cycle through a repetitive countdown to the next connection attempt:



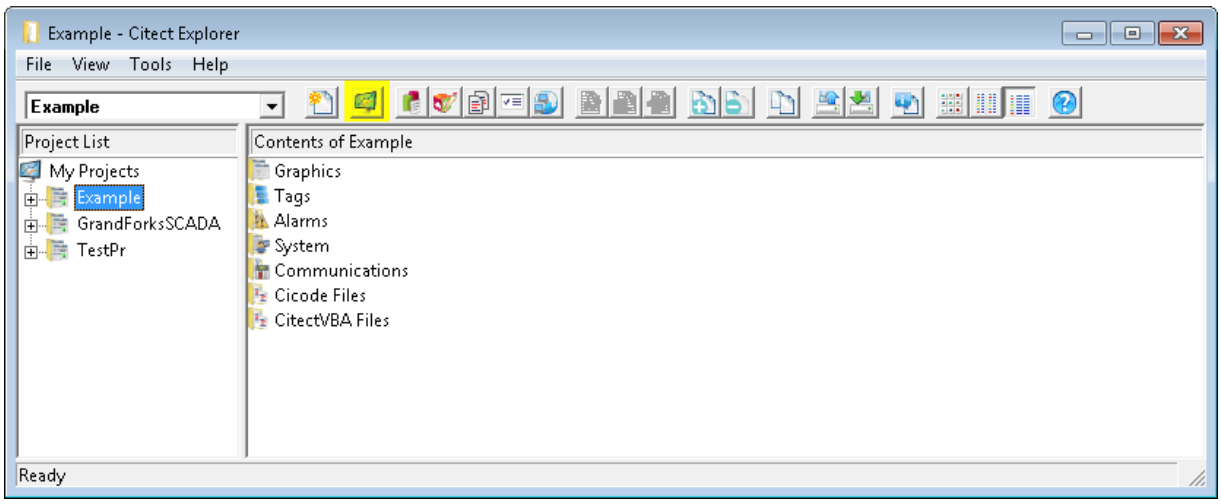
CTAPILink Not Connected

To configure the path to **CTAPI.DLL**, click on the **Search for CTAPI.DLL** menu item to open the following window:



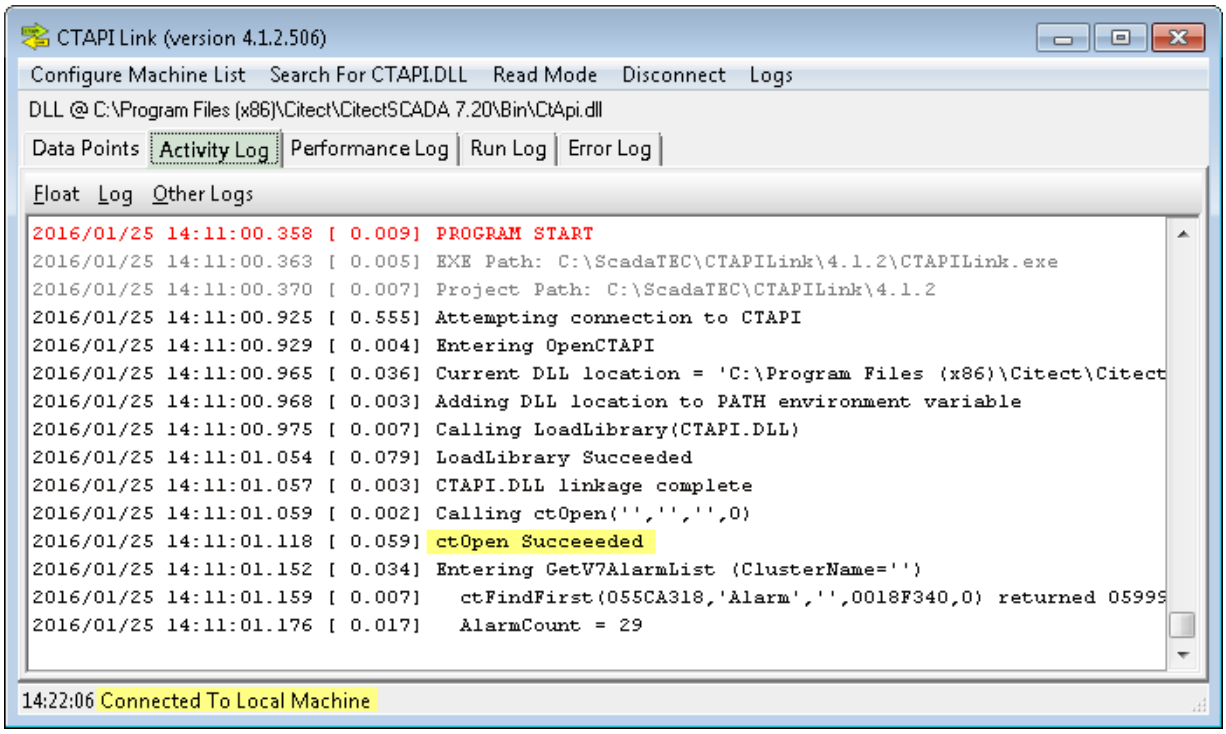
Search for Citect SCADA's CTAPI.DLL & Select a CTAPI.DLL that matches Citect SCADA Version

After the proper **CTAPI.DLL path** has been configured, launch **Citect SCADA Runtime**. The next screen assumes that the **Example** project that comes with **Citect SCADA** is running



Launch Citect SCADA Runtime

After **Citect Runtime** has launched, check the status bar at the bottom of **CTAPILink**'s main window; it should read "**Connected To Local Machine**":

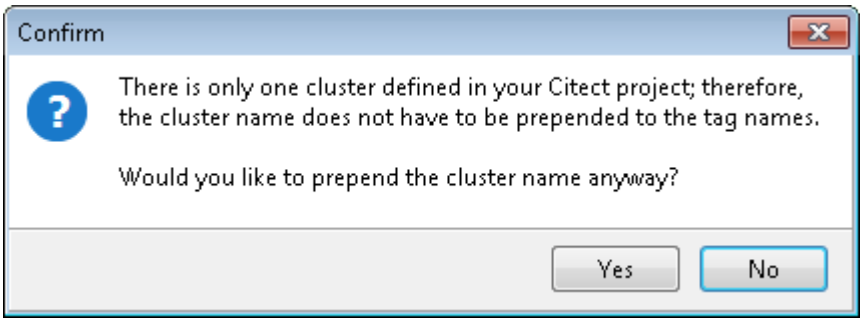


Verify that CTAPILink is Connected to Local Citect SCADA

After the initial setup, manually launching **CTAPILink** will not be necessary; the **CTAPILink Install program** inserts the proper OLE Server linkage into the Windows Registry and whenever a program such as ScadaPhone requests a connection to **CTAPILink**, Windows will launch it automatically.

If you left ScadaPhone paused on the **CTAPILink.Open Unsuccessful** window shown previously in this application note, you can now click the **Retry** button; otherwise, repeat the process of clicking the **Browse** menu item from ScadaPhone's **Alarms** tab and selecting **ScadaLink connection: Citect SCADA** from the Alarm Browse Source Selector.

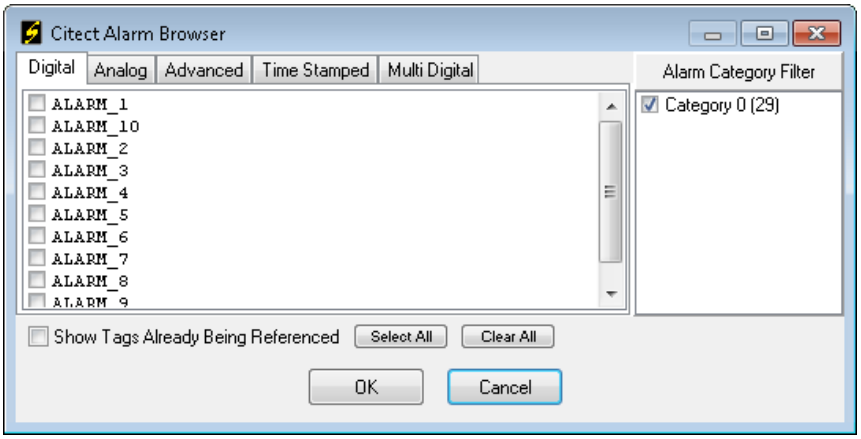
After ScadaPhone successfully obtains the **alarm information** from **Citect** via **CTAPILink**, and depending upon your project's utilization of **Citect Clusters**, you may be presented with the following prompt:



Citect Cluster Prompt

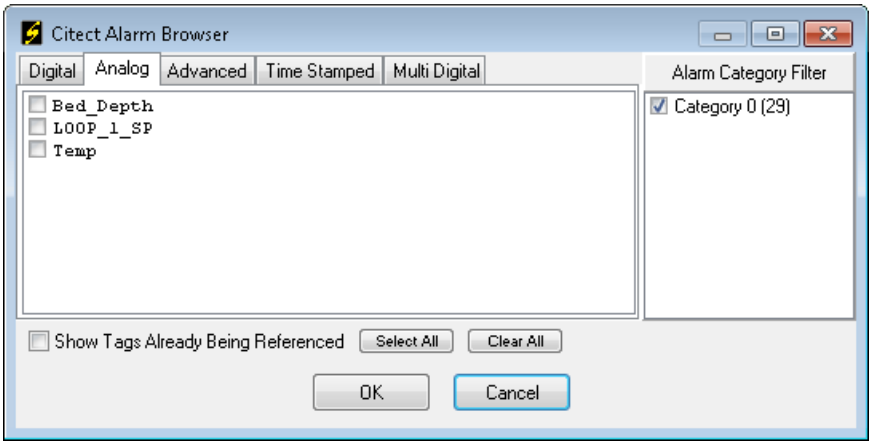
In this example, this window has been triggered because the **Example** project only has one cluster defined; therefore, ScadaPhone could obtain tag values by simply using the tag name. Citect projects containing **multiple clusters** require ScadaPhone to reference tags by **cluster name plus tag name** (e.g. **Cluster1.AnalogTag1**). This prompt is allowing you to choose whether or not ScadaPhone should use the **Cluster Name**.

If you are building a project in phases, and you know that there will be additional clusters added at a future date, choose **Yes**; this makes ScadaPhone project expansion much easier when future clusters are added to your Citect project. If you know that there will never be any other clusters, you can choose **No**. When browsing the **Example**, choose **No**. You should now see the **Citect Alarm Browser** window:



Citect Alarm Browser (Digital Tags Tab)

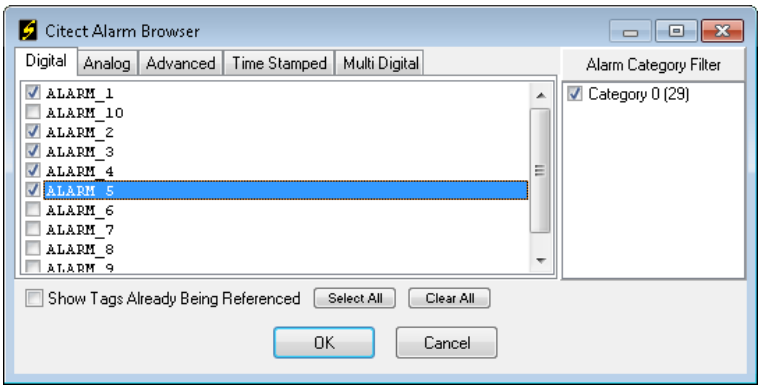
Note that the **Citect Alarm Browser** has individual tabs to segregate the different types of alarms available in Citect. In the previous image, the **Digital** tab is selected, so the list contains only **Digital** type alarms. To see the other types of alarms, click the appropriate tab:



Citect Alarm Browser (Analog Tags Tab)

Note that the list of **Digital** alarms has been replaced by the three **Analog** alarms defined in the **Example** project. Note that the **Alarm Category Filter** on the right side of the browser shows that there is only one **Alarm Category** defined in the **Example** project (**Category 0**). Larger projects with multiple categories, removing a check-mark (✓) from a low-priority category of alarms will narrow the selection list and facilitate the browsing process. Note that the number in parenthesis after the category name indicates the number of alarms defined in each category; in this example, there are **29** alarms in **Category 0**.

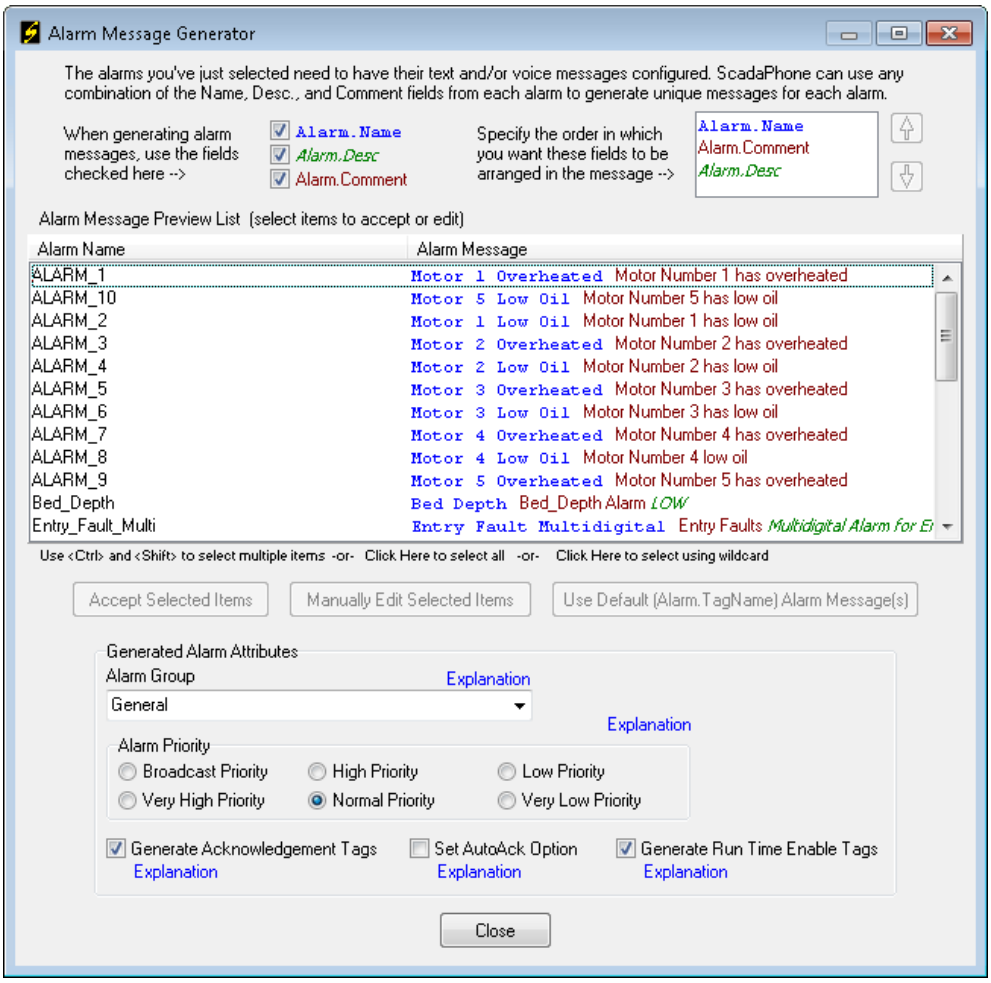
To select alarms that you want to include in your ScadaPhone project, simply put a check-mark next to the alarm names in the selected list:



Selecting Alarms to be incorporated into ScadaPhone

After all of the desired alarms have been selected with check-marks, clicking the **OK** button at the bottom of the **Citect Alarm Browser** window will move the browsing process to the next phase:

Alarm Message Generation:



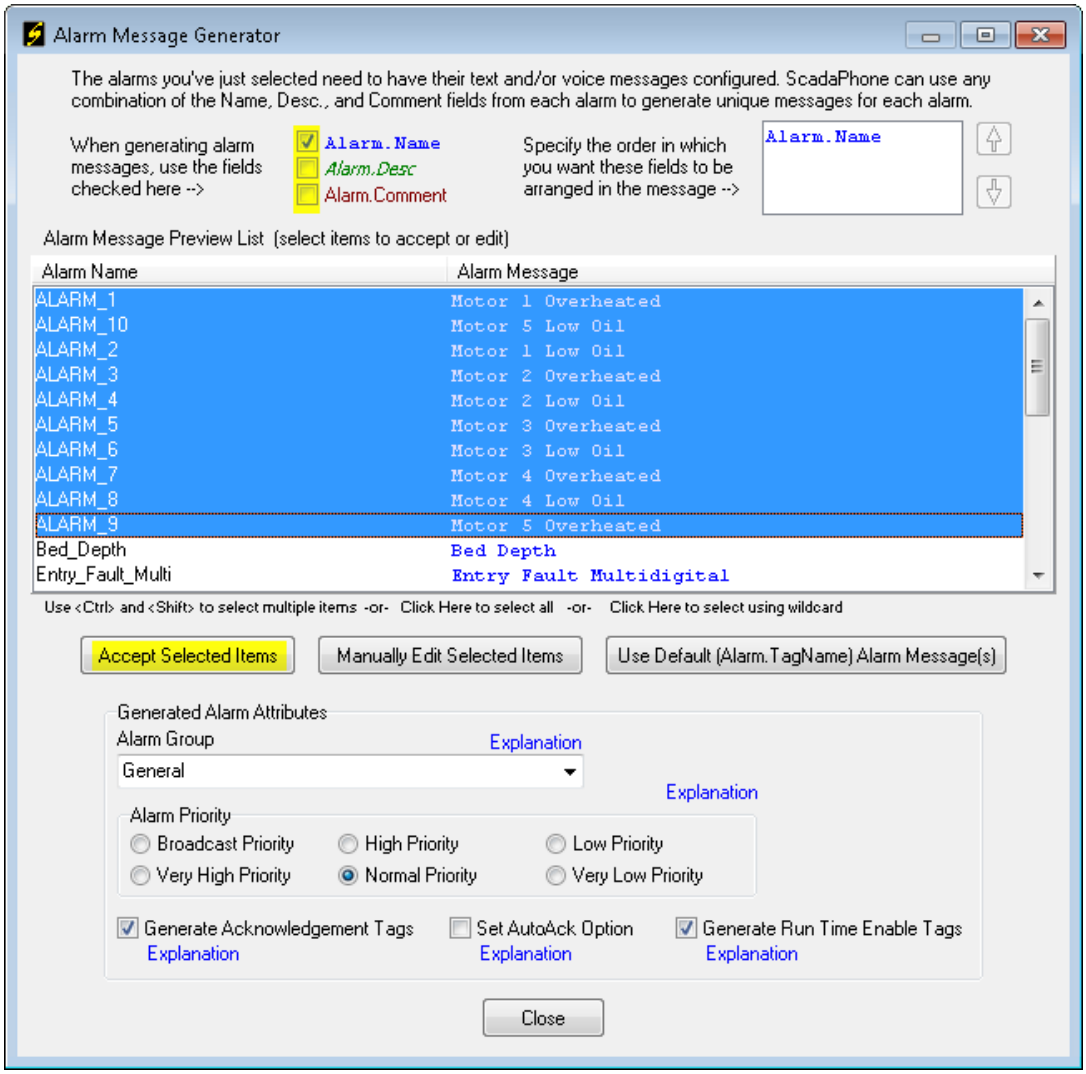
Alarm Message Generator – All 3 Fields Selected

The **Alarm Message Generator** provides the means for selecting the messages used by ScadaPhone The process will generate both a SMS message for each alarm and a Voice Wav file that will be used to announce the alarm. In well-defined projects, a suitable message can usually be found in the **Alarm.Desc** or **Alarm.Comment** field. The **Alarm Message Generator** makes an attempt to identify which field has the most verbose and unique messages; if there is a clear choice, the alarm field check marks and field sequence list will be set accordingly.

The **Example** project provides a good example of how to use the field sorting and filtering features.

At the beginning of the message-generation process, all 3 field-selection check-marks are selected (which results in some awkwardly-worded and redundant alarm messages). So, the next step is to set the field-selection check-boxes to see which pattern yields the best messages.

In the previous image, **ALARM_1** through **ALARM_10** have acceptable messages in the **Alarm.Name** field (in blue font); therefore, removing the check-marks from the **Alarm.Desc** and **Alarm.Comment** boxes yields acceptable messages for these alarms:

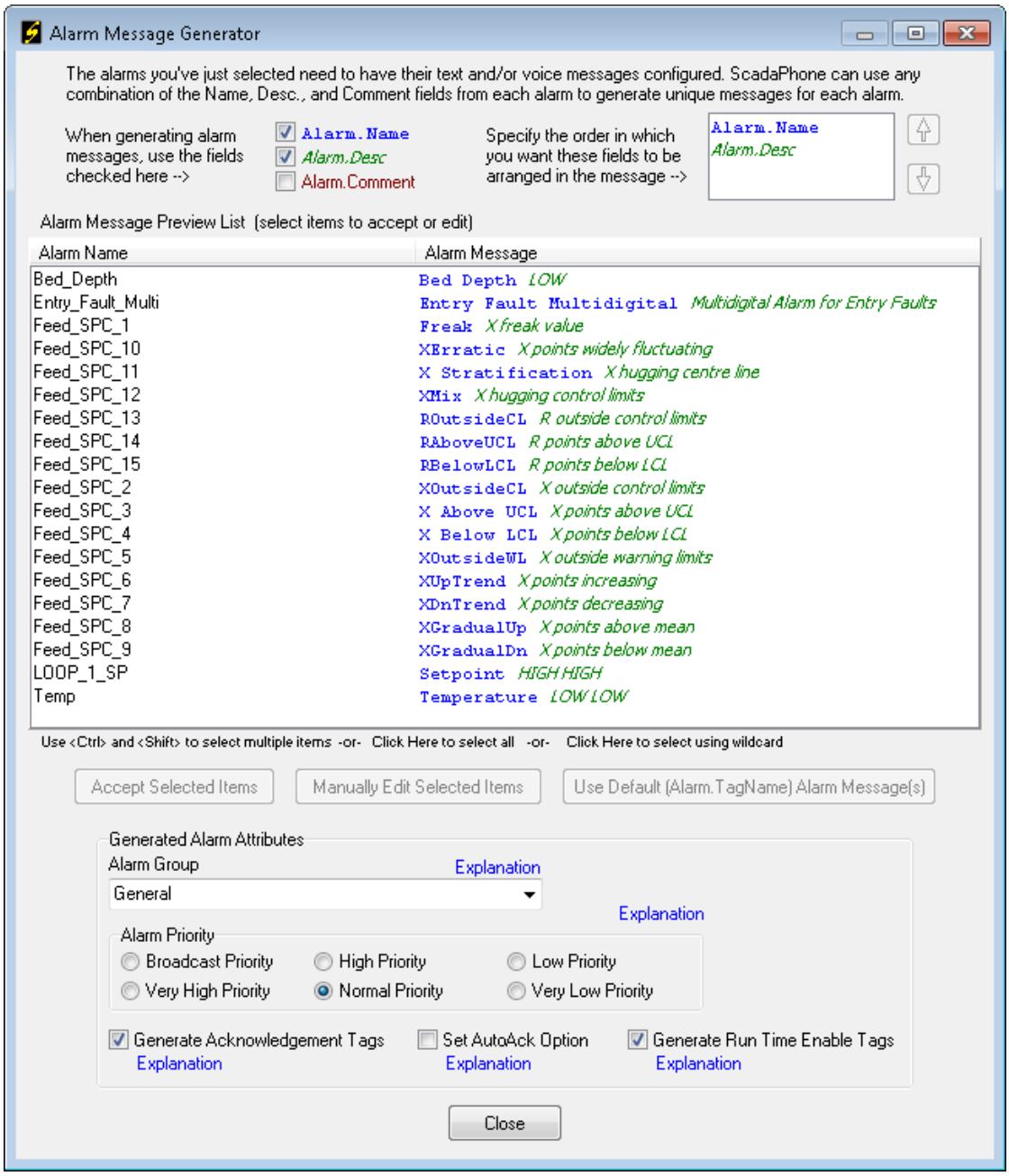


Better Formatted Alarm Messages

This selection has produced 10 suitably-worded alarm messages which are shown highlighted above. Clicking **Accept Selected Items** will add them to the ScadaPhone project, remove them from the **Alarm Message Generator** list and reevaluate the field suggestions for the remaining items.

The **Generated Alarm Attributes** are applied to each accepted alarm. The recommended settings are selected by default and there is really no need to change these settings during this process.

In the remaining items, note that the **Alarm.Comment** field has not been set in the **Example** project, so the check-mark in the **Alarm.Comment** field-selector has been omitted. The remaining fields still produce a list of awkward and redundant messages:

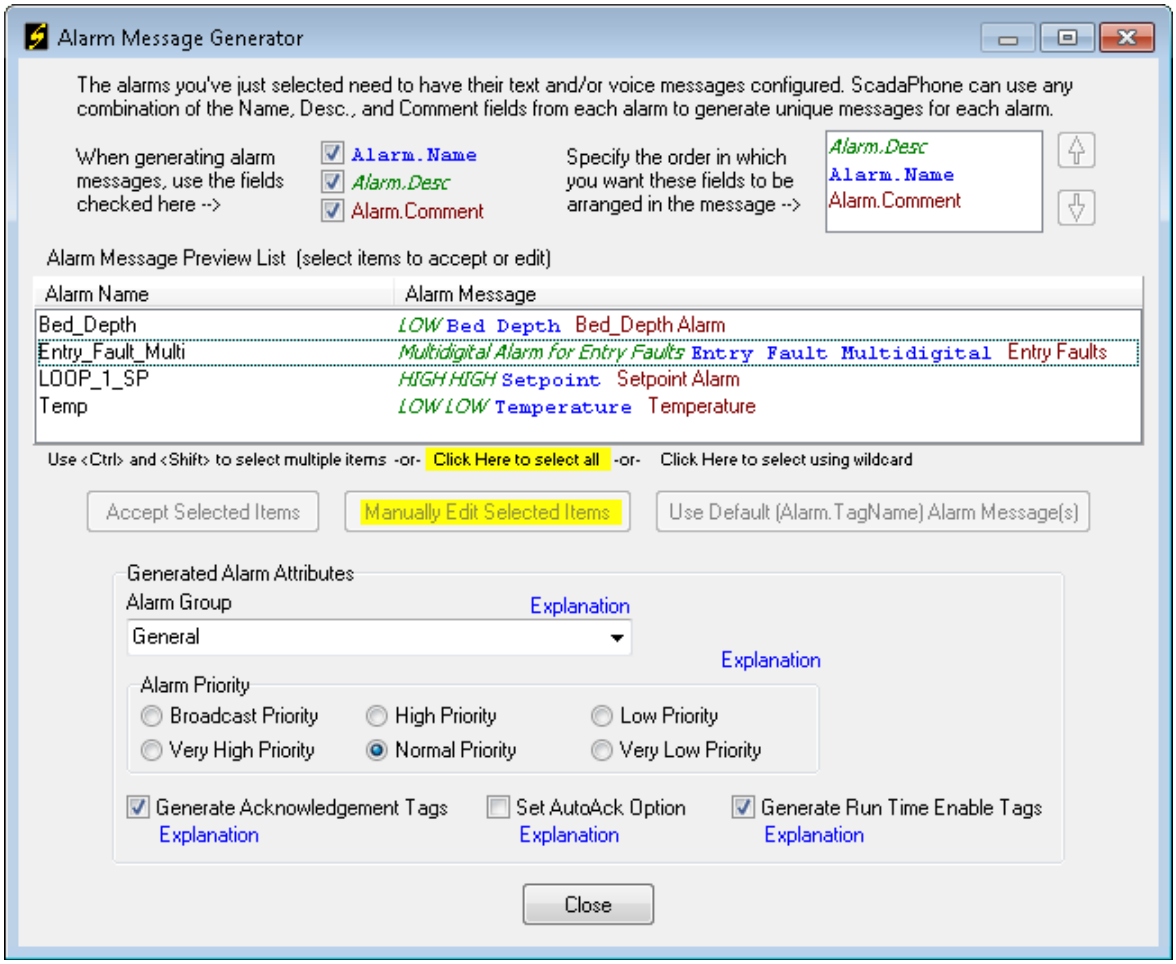


Alarm Message Generator – 2 Fields Selected

The **Feed_SPC_*** alarms look as though the **Alarm.Desc** fields (in green font) would make the most descriptive alarm messages, so the best action at this point would be to remove the check-mark from the **Alarm.Name** field-selector, highlight all of the **Feed_SPC_*** items, and click the **Accept Selected Items** button.

Manually Editing Alarm Messages:

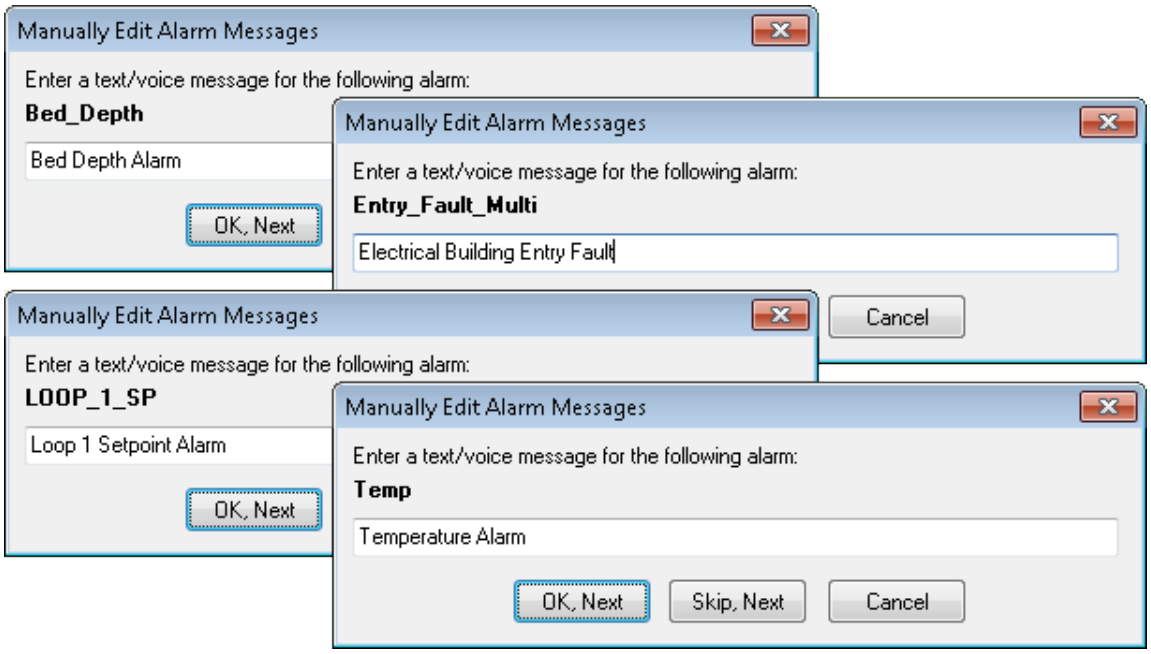
After the **Feed_SPC_*** alarms have been accepted, the only items remaining from the **Example** project are **3 Analog** alarms and **1 Multi-Digital** alarm:



Small Number of Alarm Messages Can be Manually Edited

In this group, the **Alarm.Comment** field (brown font) appears to be the best selection; however, the brown-text captions could be improved upon with some minor manual editing.

To manually edit the remaining messages, click the small label reading **Click Here to select all** and then click the **Manually Edit Selected Items** button.



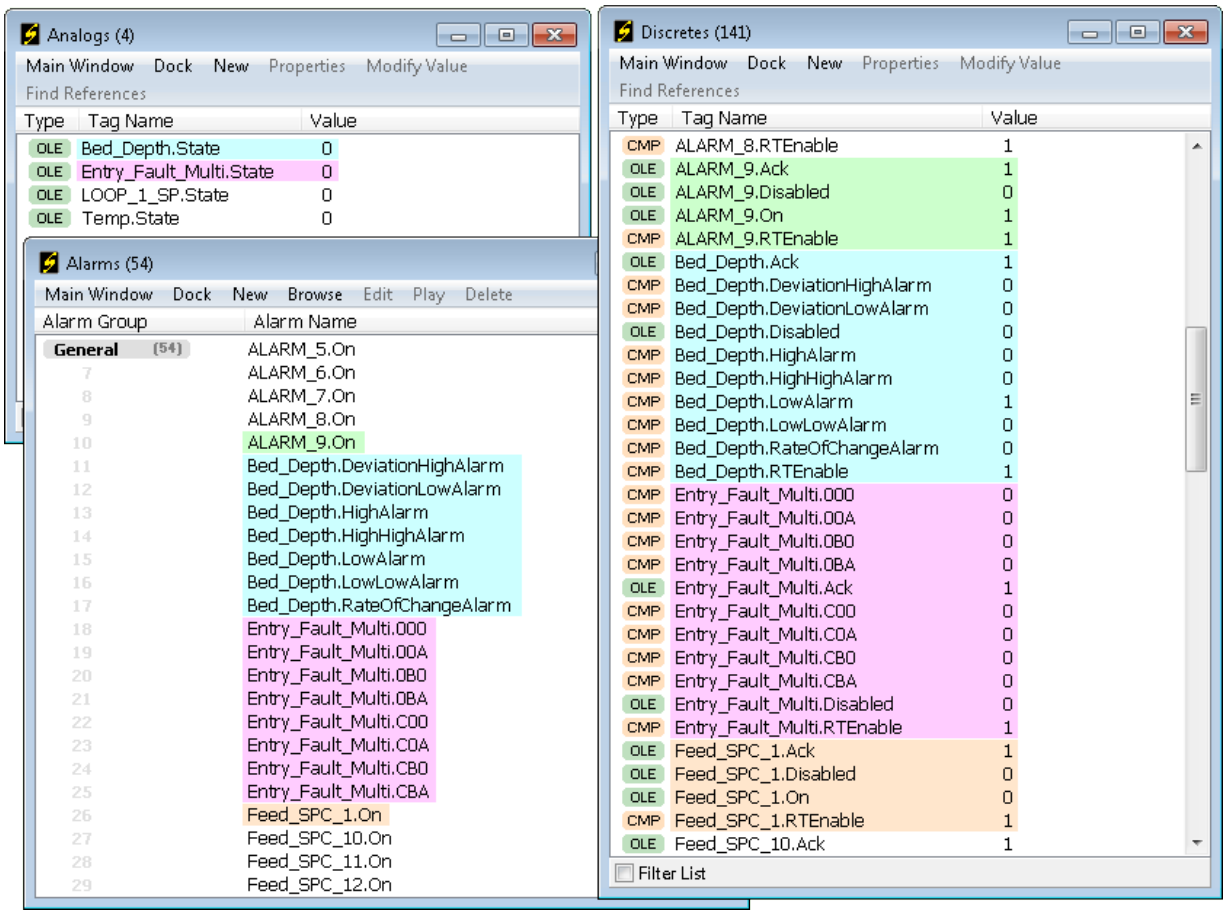
Manually Edit Alarm Messages

So far this document has described the **Accept Selected Items** and **Manually Edit Selected Items** methods for setting alarm messages in alarms obtained from a Citect CTAPI Alarm Browser.

The third and final option for setting the alarm messages is the **Use Default (Alarm.TagName) Alarm Message(s)** button; this is the most simplistic way to accept alarms browsed from Citect, but the drawback to using this method is that the generated alarm messages are not as descriptive as can be derived from the other acceptance methods.

After all of the items in the **Alarm Message Preview List** have been accepted and the list is empty, the **Alarm Message Generator** window will *automatically close* and focus will return to the main window. Note that there is not a 1-to-1 correspondence between the number of Citect alarms accepted and the number of ScadaPhone alarms produced; **Analog** and **Multi-Digital** Citect alarms are supported in ScadaPhone via the use of **Computed Tags** which produce **Discrete** bit-values by masking status bits out of the Citect **Alarm.State** field. The actual alarm-and-tag count produced is as follows:

- **Digital**, **Advanced**, and **Time-Stamped** alarms produce: **1** discrete alarm and **4** discrete tags
- **Analog** alarms produce: **7** discrete alarms, **10** discrete tags and **1** analog tag
- **Multi-Digital** alarms produce: **8** discrete alarms, **11** discrete tags and **1** analog tag

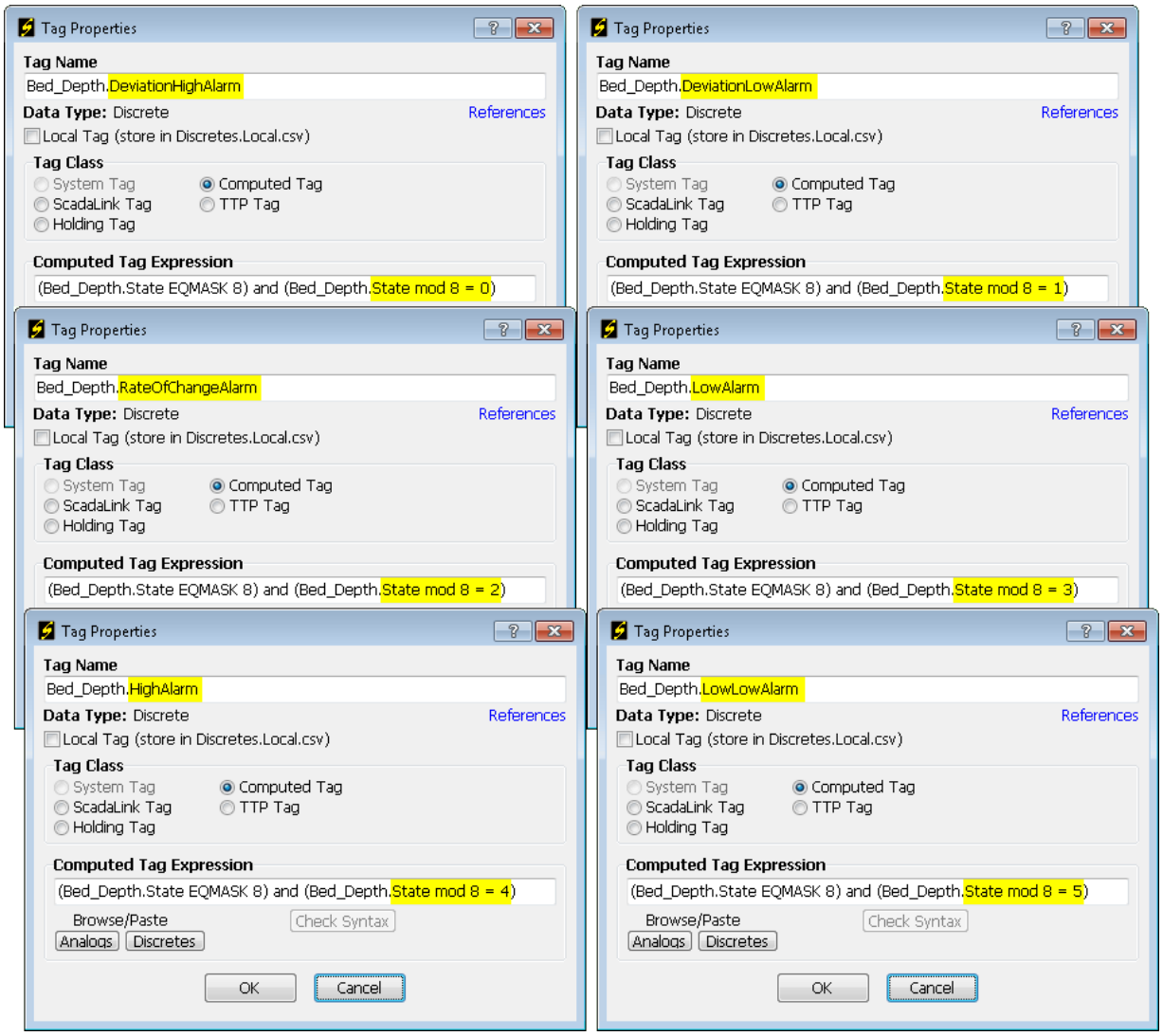


Alarm Browse Results

This scheme does increase the tag-count, but it does a very good job of tracking the ***actual*** Citect **Alarm.State** status.

Another reason for splitting the **Citect Analog** and **Multi-Digital** alarms into multiple **Discrete ScadaPhone** alarms is compatibility: Citect implements some alarm types that ScadaPhone does not *directly* support; ScadaPhone's **Computed Tags** feature provides a means to bridge this gap.

To see how this is implemented, click the **Discretes** tab, highlight any of the Computed Tags associated with any Analog or Multi-Digital Alarm:



Computed Tags

The computed tags driving the **Citect Analog/ScadaPhone** Discrete alarm bits use the bit masked value maintained in the Citect **Alarm.State** field. When the alarm is active in Citect, the **4th bit** in the **State** bit mask is a '**1**'; ScadaPhone implements a discrete-logic operator called **EQMASK** to test bit-statuses at runtime. The **EQMASK** operator takes two analog values as input arguments and returns a discrete result: **<RESULT> = <ARG1> EQMASK <ARG2>**.

Note that ScadaPhone analogs are all stored as ***floating-point*** numbers; therefore, when evaluating expressions that require ***integer*** inputs (such as **EQMASK**, **DIV**, and **MOD**), the floating point values are ***rounded*** to the nearest **32-bit integer** before the integer-based operation is performed.

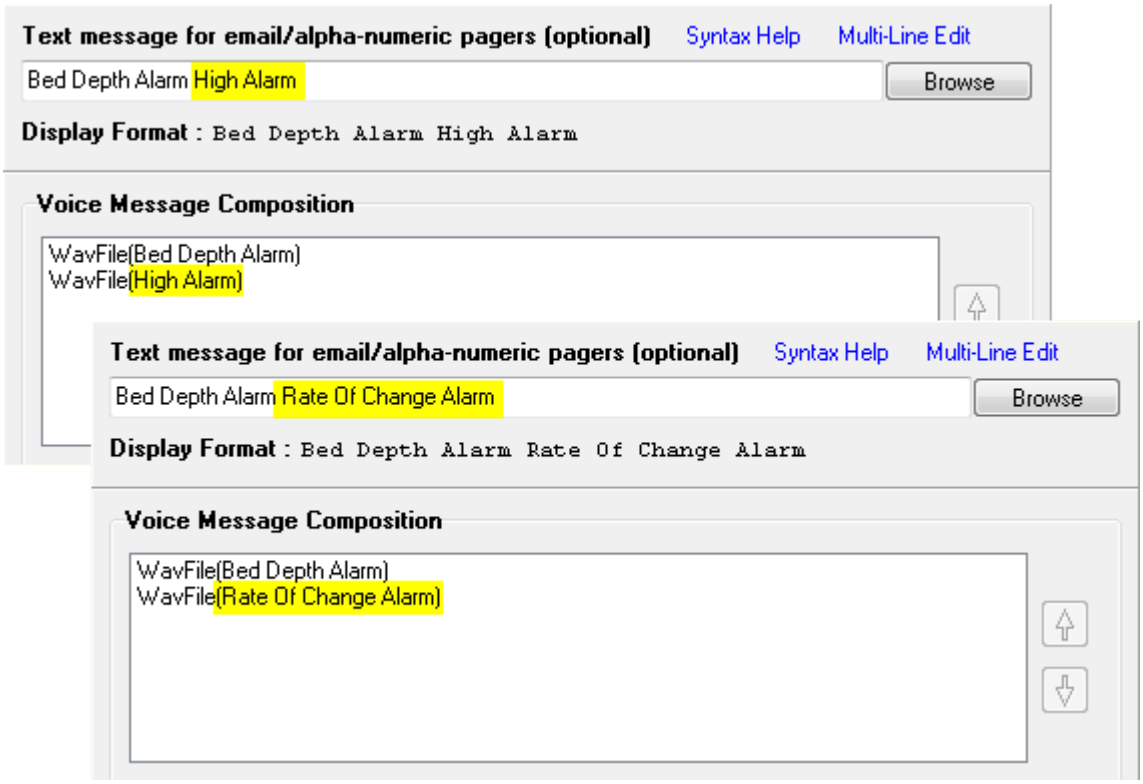
In the previous image of **Tag Properties**, the first part of each **AND** expression is checking to see if **bit 4** is a **1** (meaning the alarm is ***active*** in Citect).

The second half of each computed tag expression, **(Alarm.State mod 8 = *x*)**, narrows down the ***type*** of alarm that is active.

The Citect **Alarm.State** bit-mask uses the last **3** bits to specify the type of alarm. ScadaPhone recognizes 7 different Citect Analog Alarm types:

```
(Alarm.State EQMASK 8) and (Alarm.State mod 8 = 0) Deviation High
(Alarm.State EQMASK 8) and (Alarm.State mod 8 = 1) Deviation Low
(Alarm.State EQMASK 8) and (Alarm.State mod 8 = 2) Rate of change
(Alarm.State EQMASK 8) and (Alarm.State mod 8 = 3) Low
(Alarm.State EQMASK 8) and (Alarm.State mod 8 = 4) High
(Alarm.State EQMASK 8) and (Alarm.State mod 8 = 5) LowLow
(Alarm.State EQMASK 8) and (Alarm.State mod 8 = 6) HighHigh
```

The 7 specific alarm states are differentiated in the ScadaPhone alarm messages by using the **alarm message** accepted in the **Alarm Message Generator** as a ***message prefix*** and the specific state as a ***message suffix***:

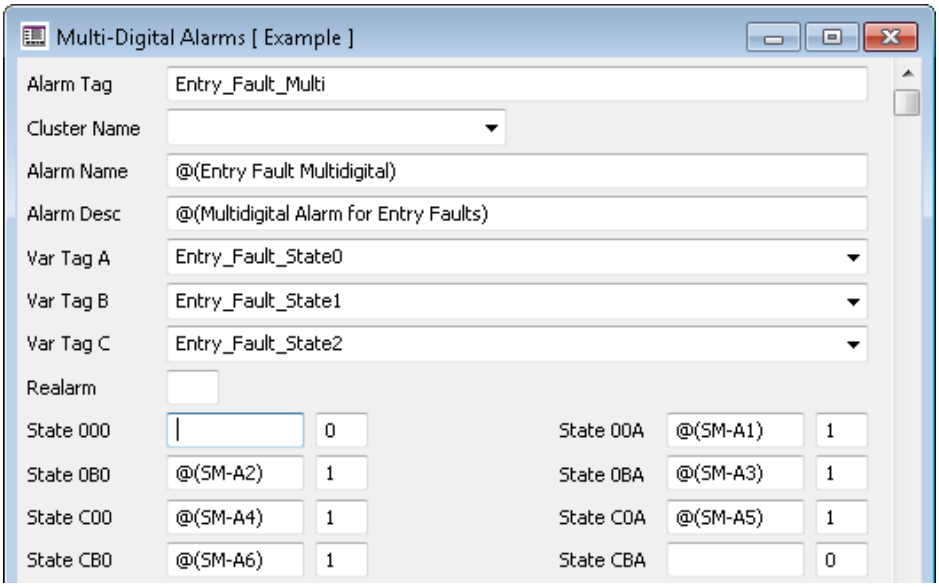


Generating Text & Voice Messages

Citect's Multi-Digital alarms are implemented similarly to the Citect Analog alarms; each Citect state is identified by using bit-mask operators on the **Alarm.State** field:

```
(Alarm.State EQMASK 2048) and (Alarm.State mod 8 = 0) State 000
(Alarm.State EQMASK 2048) and (Alarm.State mod 8 = 1) State 00A
(Alarm.State EQMASK 2048) and (Alarm.State mod 8 = 2) State 0B0
(Alarm.State EQMASK 2048) and (Alarm.State mod 8 = 3) State 0BA
(Alarm.State EQMASK 2048) and (Alarm.State mod 8 = 4) State C00
(Alarm.State EQMASK 2048) and (Alarm.State mod 8 = 5) State C0A
(Alarm.State EQMASK 2048) and (Alarm.State mod 8 = 6) State CB0
(Alarm.State EQMASK 2048) and (Alarm.State mod 8 = 7) State CBA
```

The A, B, and C states refer to the bit pattern created by the **Var Tag A**, **Var Tag B**, and **Var Tag C** inputs to the **Citect Multi Digital** alarm. For example, the **Entry_Fault_Multi** alarm browsed from Citect's Example project is defined in the Citect Project Editor as follows:

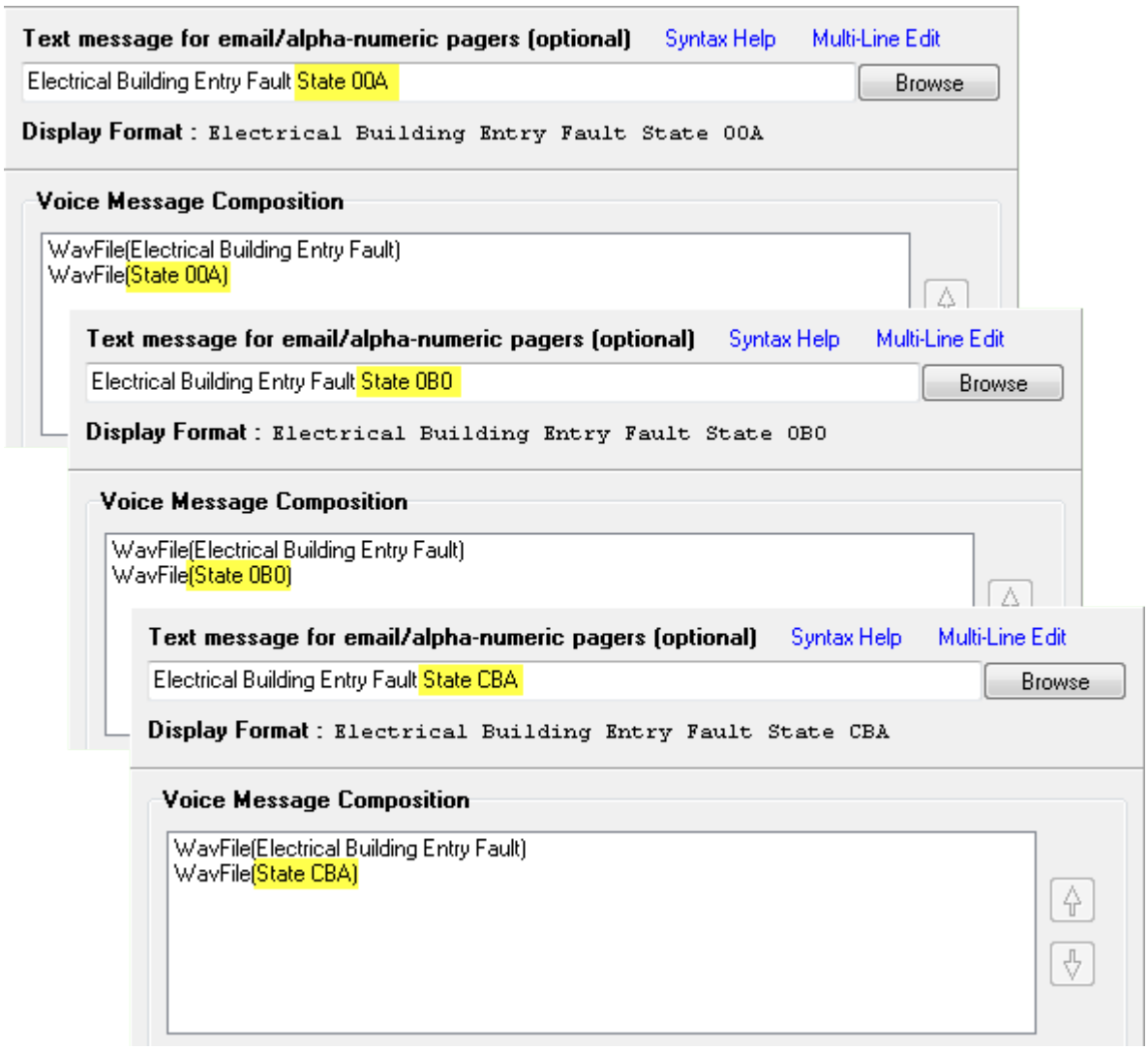


This setup corresponds to the following table:

	Var Tag A Entry_Fault_State0	**Var Tag B** Entry_Fault_State1	**Var Tag C** Entry_Fault_State2
000	0	0	0
00A	1	0	0
0B0	0	1	0
0BA	1	1	0
C00	0	0	1
C0A	1	0	1
CB0	0	1	1
CBA	1	1	1

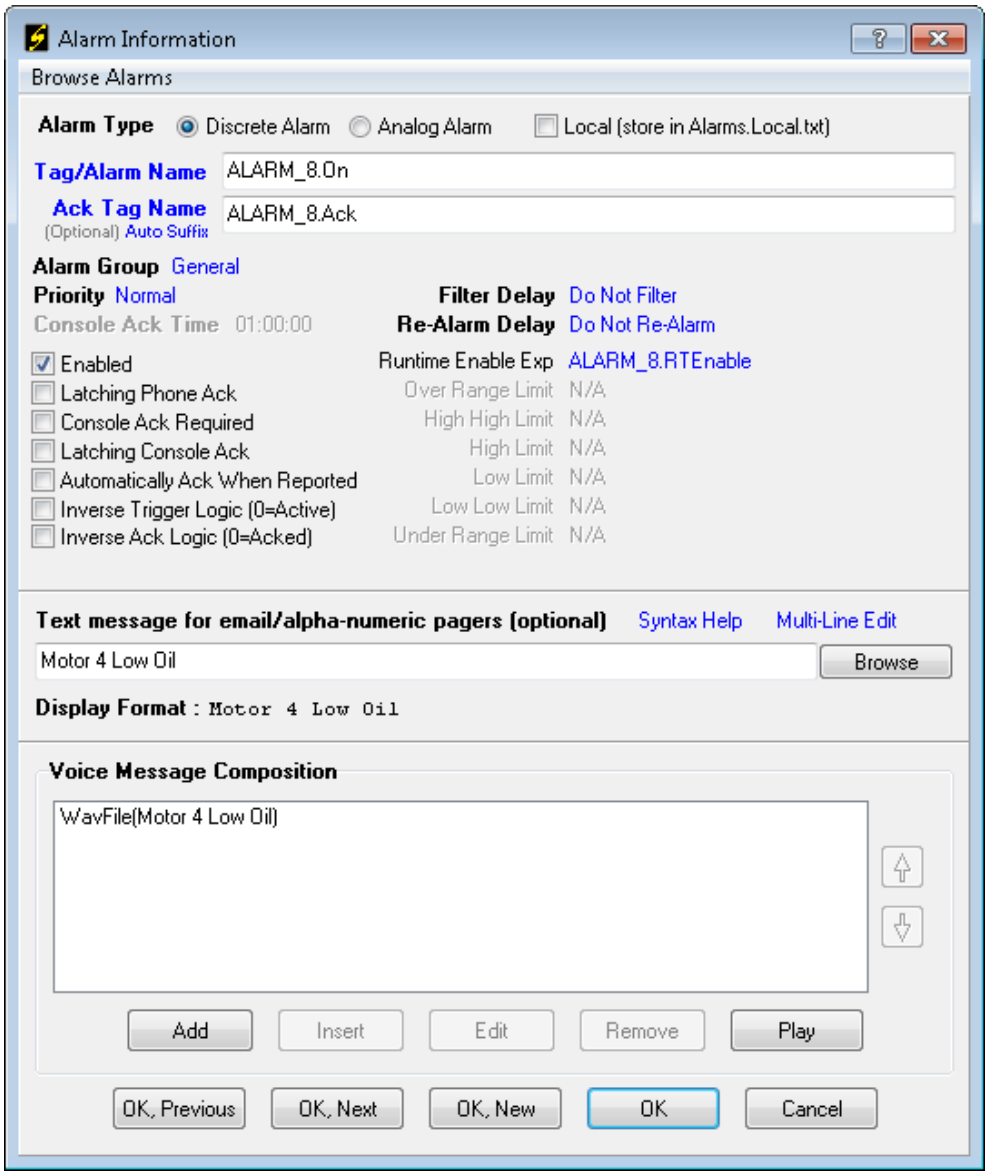
In ScadaPhone, a separate Discrete alarm is generated for each possible state even though each state might not represent an alarm state; however, there should be no false alarms reported because of the first condition in the **Multi-Digital** alarm computed tag: **(Alarm.State EQMASK 2048)**. 2048 = 100000000000 in binary (which is **bit 11**). If the current state is *not considered to be an alarm in Citect*, **bit 11** will be a '**0**', so the computed tag expression driving the alarm will return **FALSE**.

ScadaPhone's **Citect Alarm Browser** ensures that each **Discrete** alarm in ScadaPhone identifies the **Citect "CBA" state** in both the **Text** and **Voice** alarm messages:



Text and Voice Messages from Multi-Digital Alarms

All other Citect alarms are simply implemented as discrete alarms which are triggered by the value of the **Alarm.On** field obtained from Citect at runtime:



Typical Generated Alarm Configuration

Note that the message "***Motor 4 Low Oil***" which was accepted in the **Alarm Message Generator** has been inserted into both the **Text Message** and the **Voice Message** for this alarm. Also note that the **Ack Tag** was also automatically configured by the browser (because the **Generate Acknowledgement Tags** option was selected in the **Alarm Message Generator** when the first batch of digital alarms were accepted.

ScadaPhone's Citect Alarm Browser can save many hours of configuration time when **ScadaPhone** is being used in conjunction with ***large* Citect** projects.