Table of Contents

Version: 1.0 / 2022-03-31

Introduction

This document describes the integration of Audit Logging for applications built with VisionX. The Audit Log Add-on offers historization for database tables. Therefore, all data changes are completely logged according to auditability including time, user and the data that has been changed.

Installation

To install the Add-on, launch VisionX and click “Ready Made Solutions” at the bottom of the screen.

Then click on “AddOns” on the left side of the screen under “Categories” and click on the orange plus icon to install it. A restart of VisionX is required after the installation is complete.

After restarting VisionX, enable the module for an existing application by opening the application and selecting “Modules” in the VisionX settings menu:

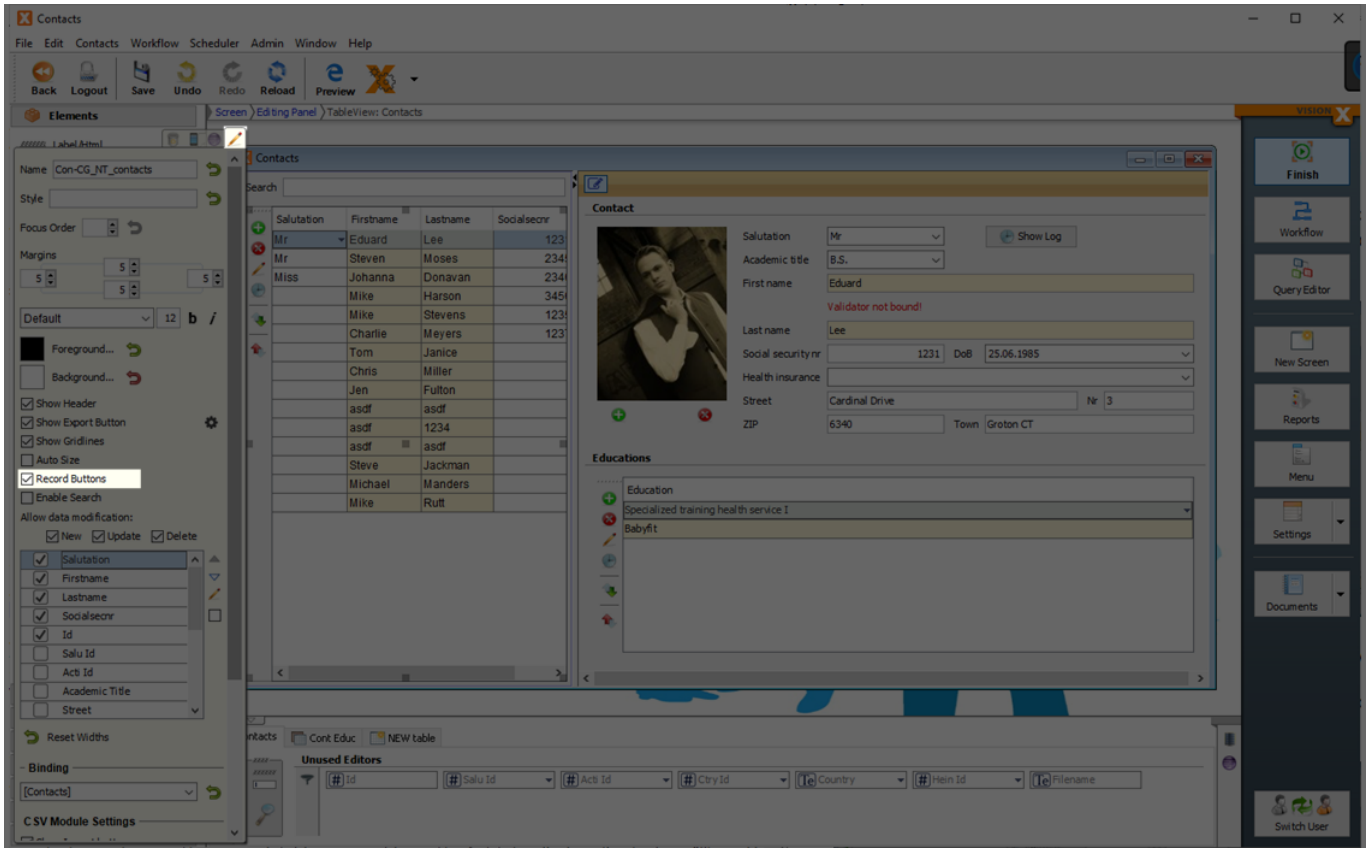
Select “AuditLog” under “Standard Modules” and check the box in the “Installed” column:

If you create a new application, the module is enabled automatically.

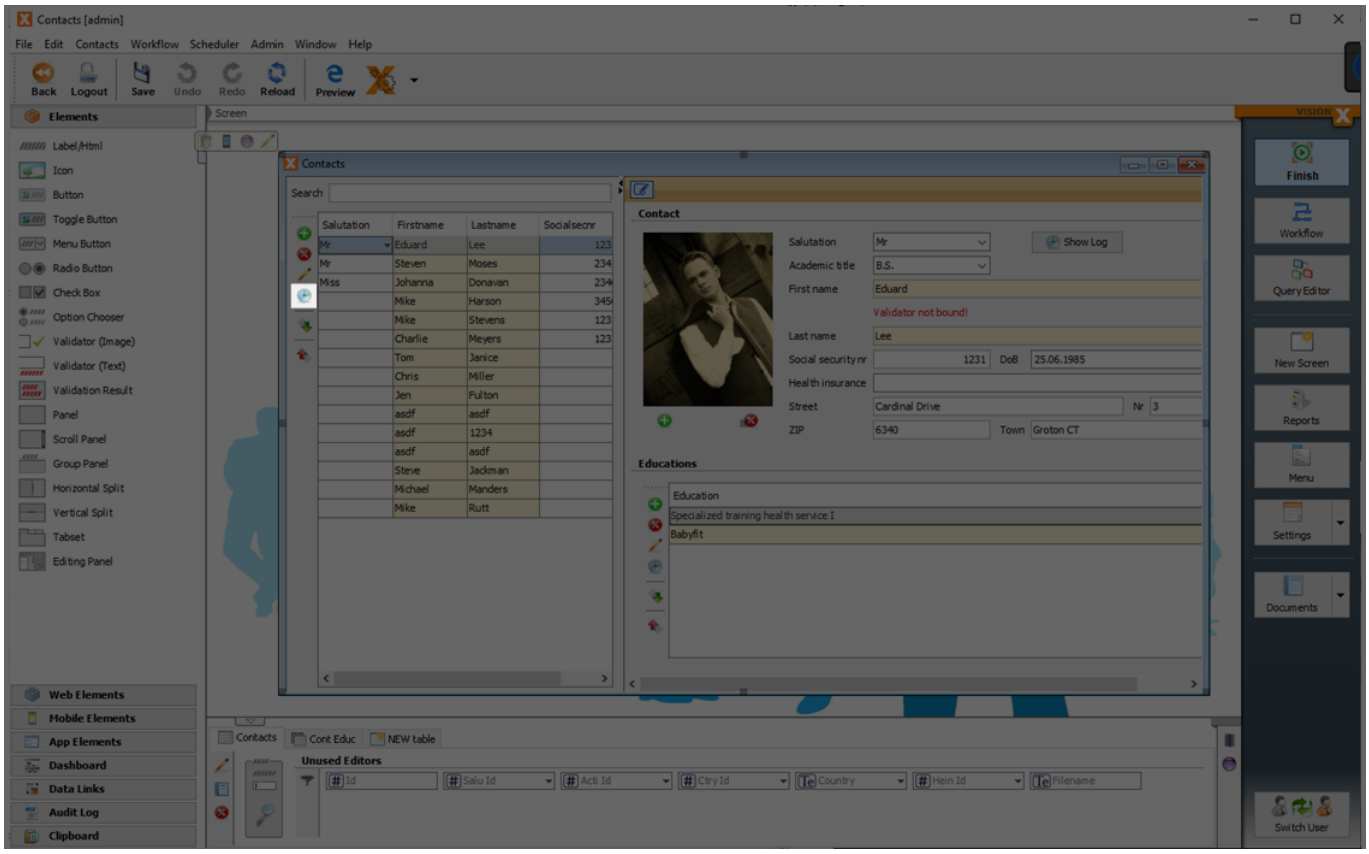
Audit Log

After the module is enabled for an application, the Audit Log button is shown on the navigation panel for every table.

Note: On the web, mobile and universal screen layouts, the navigation panel is hidden by default. To show it, select the table in design mode, click on the pencil icon to open the customizer, and check the “Record Buttons” box.



Click on this button to show the Audit Log Dialog.



All logged operations for the currently selected row in the original table are shown by default.

If you check “Show all records”, the logged operations for all records from the table are shown no

matter which row is selected in the original table.

By default, only the columns shown in the original table are visible in the Audit Log Dialog. By checking “Show all columns”, all the columns of the original table can be viewed.

You can also filter the data shown in the Audit Log Dialog.

“Logged between” filters all the data that has been changed in the specified time window. “Show changes for column” filters all rows in which the specified column has been changed. Applying the “Type” filter only shows log changes of a specific type (insert/update/…).

In general, all changed data for a row is marked bold in the Audit Log Dialog. This can easily be seen in the green “Insert” rows as every column in a newly inserted row has been “changed”.

Any record can be restored using the “Restore” button.

If you click on the restore button, the selected row is inserted/updated in the original table. The values are taken from the log table. If a row with the same primary key value is found in the original table, the row is updated; otherwise a new row with the values from the log table is inserted.

If the restore finished successfully, the Audit Log Dialog remains open and the inserted/updated row can be seen in the original table. The restored row is colored light blue in the Audit Log Dialog.

Possible Log Change states are:

- Inserted (green)
- Updated (yellow)
- Deleted (red)
- Restored (blue)
- Restored with error (red)

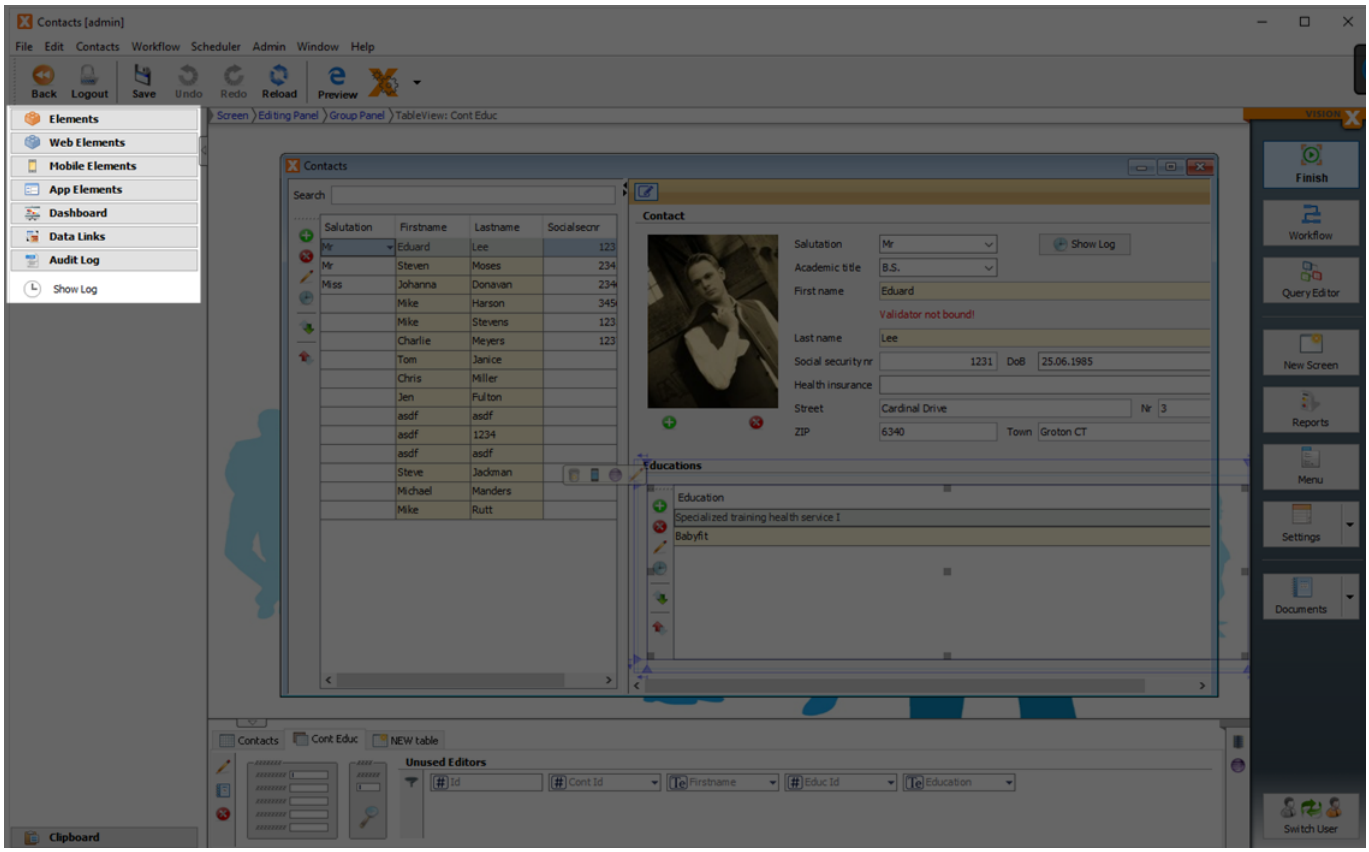
If the restoration of a row cannot be finished successfully, an error message is shown and the Audit Log Dialog is closed. So after checking the insert failure reason, you can immediately edit the new corrupted row in the original table, correct it and save it.

Sometimes we want to log tables that contain binary data such as images. Usually the Audit Log copies the whole row into its specific log table. However, binary data can take up huge amounts of space in your database. Therefore the Audit Log handles tables with binary data a bit differently. The binary data is not stored directly in the log table, but instead in a table called LOG_BINARIES. The log table only stores a reference to this LOG_BINARIES table in its rows. If the binary data in the original table is changed, a new row is created in the LOG_BINARIES table and then referenced in the log table.

The binary data can be downloaded via the download button in the Audit Log Dialog.

Audit Log Button

In addition to the default button that is shown in the table navigation panel, you can also add your own button to open the Audit Log Dialog. This button is available on the list of UI elements in the VisionX editor after you activated the AuditLog AddOn.



After you have dragged the “Show Log” button into the desired location on your screen, it is automatically bound to the nearest DataBook. You can also bind this button to a specific column in your table using the button customizer by selecting the button and clicking on the pencil icon.

The bound column is automatically selected in “Show changes for column” in the Audit Log Dialog.

Audit Log Util

If you want to go a step further and configure the Audit Log more specifically, you can use the AuditLogUtil class. This class implements a lot of functionality to configure it to your liking.

The following Functions are available:

```
/**
 * Sets if the Audit Log button is shown by default for all the components
 * in the specified container.
 *
 * @param pContainer the container
 * @param pShowTableAuditLogButton if the AuditLog button should be shown by
default.
*/
setDefaultShowTableAuditLogButton(IContainer pContainer, boolean
pShowTableAuditLogButton)
```

```
/**
 * Clears the default settings of showing the Audit Log button for the
container.
 *
 * @param pContainer The container.
*/
public static void clearDefaultShowTableAuditLogButton(IContainer
pContainer)
```

```
/**
 * Sets if the AuditLog button should be shown for this specific component.
 *
 * @param pComponent the component
 * @param pShowTableAuditLogButton if the AuditLog button should be shown.
*/
public static void setShowTableAuditLogButton(IComponent pComponent, boolean
pShowTableAuditLogButton)
```

```
/**
 * Clears the setting if the AuditLog button should be shown for this
specific component.
 *
 * @param pComponent the component
*/
public static void clearShowTableAuditLogButton(IComponent pComponent)
```

```
/**
 * Sets if the export button is shown by default in the Audit Log Dialog
 * for all the components in the specified container.
 *
 * @param pContainer The container.
 * @param pShowTableExportButton if the export button should be shown by
default.
*/
public static void setDefaultShowTableExportButton(IContainer pContainer,
boolean pShowTableExportButton)
```

```
/**
 * Clears the default settings of showing the export button for the
```

```
container.
 *
 * @param pContainer The container.
*/
public static void clearDefaultShowTableExportButton(IContainer pContainer)
```

```
/**
 * Sets if the export button should be shown for this specific component.
 *
 * @param pComponent the component
 * @param pShowTableExportButton if the export button should be shown.
*/
public static void setShowTableExportButton(IComponent pComponent, boolean
pShowTableExportButton)
```

```
/**
 * Clears the setting if the export button should be shown for this specific
component.
 *
 * @param pComponent The component.
*/
public static void clearShowTableExportButton(IComponent pComponent)
```

```
/**
 * Sets if the restore button is shown by default in the Audit Log Dialog
 * for all the components in the specified container.
 *
 * @param pContainer The container.
 * @param pShowTableRestoreButton if the restore button should be shown by
default.
*/
public static void setDefaultShowTableRestoreButton(IContainer pContainer,
boolean pShowTableRestoreButton)
```

```
/**
 * Clears the default settings of showing the restore button for the
container.
 *
 * @param pContainer The container.
*/
public static void clearDefaultShowTableRestoreButton(IContainer pContainer)
```

```
/**
 * Sets if restore button should be shown for this specific component.
 *
 * @param pComponent The component.
 * @param pShowTableRestoreButton if AuditLog button should be shown.
*/
public static void setShowTableRestoreButton(IComponent pComponent, boolean
pShowTableRestoreButton)
```

```
/**
 * Clears the setting if the restore button should be shown for this
specific component.
 *
 * @param pComponent The component.
*/
public static void clearShowTableRestoreButton(IComponent pComponent)
```

```
/**
 * Sets the column view for the Audit Log log table for this specific
component.
 *
 * @param pUIComponent the UIComponent
 * @param pColumnView the log table column view.
*/
public static void setLogTableColumnView(UIComponent pUIComponent,
ColumnView pColumnView)
```

These functions can be called from anywhere on the client side of the VisionX application. This example shows how to disable the Audit Log Button application wide from an application screen:

```
AuditLogUtil.setDefaultShowTableAuditLogButton(getApplication(), false);
```

Configuration

Additional Features

There are three special features that you can configure in config.xml:

- Clean mode
- Ignore tables
- Login tracking

These features can be configured in the config.xml file of the application as follows:

```
<!-- auditlog is enabled by default, enabled attribute can be omitted -->
<auditlog enabled="true">
    <mode>clean</mode>
    <ignore>
      <!-- <table>CONTACTS</table> -->
      <table>CONT_EDUC</table>
      <!-- version shown by default, this tag can be omitted -->
      <version hidden="true"/>
    </ignore>
    <features>
      <!-- session tracking is disabled by default -->
      <login enabled="true"/>
```

```
    </features>
</auditlog>
```

If the mode is set to clean, the log table will be dropped every time you open a screen that contains the original table. This mode is mainly used for development/debugging purposes.

If the “mode” node is omitted or has a value other than “clean”, the log table is only created if it does not exist yet, or if the structure is different from the original table.

With “version” and its “hidden” attribute you can show/hide the version in the application. If you want see the version then set hidden to false or omit the whole tag.

The “ignore” node specifies which original tables should be ignored. This means the LOG_CONT_EDUC log table will not be created and the history feature for CONT_EDUC table will not be added in the GUI.

If the table LOG_CONT_EDUC already exists in the database, it will not be dropped. It will just be ignored. This means that the history feature will not be available for CONT_EDUC table in the GUI.

The “login” node and its encompassing “features” block specifies whether the logins in the application are logged or not. By default this feature is disabled. If the feature is enabled, a table LOG_AUDIT_SESSIONS is created. This table contains all the attempted logins into the application. It also tracks when the user logs out of the application manually or via timeout.