Resilient
an IBM Company

Resilient Incident Response Platform

ASP.NET INCIDENT CREATION APPLICATION INSTALLATION GUIDE v1.0

Published: May 2016

https://www.resilientsystems.com

Table of Contents

1. Introduction

This is an example website to demonstrate how you can create new incidents using the Resilient Systems REST API and ASP.net.

2. Installation

The DLLs are compiled using .Net version 4.5. If your environment uses a different version of .Net, please recompile the project or contact us for assistance.

Afterwards, you can extract all files in the WebSite folder into the root folder of an existing ASP.net enabled IIS website. Alternatively, you can create a new ASP.net website.

3. Configuration

The Web.config file is renamed to Web.config-sample to avoid accidental overwrite of the existing Web.config file. You can rename it back to Web.config if you are creating a new website, or copy its contents to the existing Web.config.

CreateIncident.aspx is the only page in the web app. Before you can open this page with a browser, you need to supply your Resilient credentials in the Web.config file.

The following table lists the Web.config values used by CreateIncident.aspx.

Setting	Required?	Description
Co3ApiUrl	Yes	Resilient REST API's base URL.
Co3UserAccount	Yes	Resilient user credential used to access the REST API. The user account must belong only to one org.
Co3UserPassword	Yes	

Co3ProxyUser	No	If you have an HTTP proxy, specify the proxy credentials to allow the web application to access the REST API. Note that you also need to add the following to your Web.config:
		`<system.net>`
		`  <defaultProxy>`
		`    <proxy usesystemdefault="true" proxyaddress="http://proxy_url" />`
		`  </defaultProxy>`
		`</system.net>`
		Be sure to replace proxy_url with the correct value.

Co3ProxyPassword	No	
Co3ProxyDomain	No	
Co3CssRequired	No	If specified, this CSS class is added to fields to indicate that they are required.

Setting	Required?	Description
Co3CssInvalid	No	If specified, this CSS class is added to fields with invalid data during postback.

4. Customization

The CreateIncident page consists of the ASPX file and its CS corresponding Code-behind. CreateIncident.cs has most of the logic to connect to the Resilient REST API with the controls on the ASPX page. You can do most if not all of the customization by editing the ASPX file. The following sections describe some of the actions you can take to adapt the page to your needs.

4.1. JavaScripts

The web application does not rely on JavaScripts to function properly. You can optionally add JavaScripts to the page to improve ease of use and page effects. CreateIncident.aspx has an example of how to add date or date time pickers to the controls using jQuery.

4.2. Master File

CreateIncident.aspx is a standalone ASPX page. If your site uses master files, you can convert CreateIncident.aspx to support master files by doing the following:

- Add the MasterPageFile attribute to the <%@Page> tag.
- Remove the <html> and <head> tags.
- Move the style and script links to the appropriate location of your website.
- Replace the <body> tag with the <asp:Content> tag.

4.3. Web Application

If your website was created as a web project, where the Code-behind files are compiled into a DLL, and you wish to incorporate this example, you can do so by editing CreateIncident.aspx and changing the CodeFile attribute in the <%@Page> element to CodeBehind. For example:

```
CodeBehind="CreateIncident.aspx.cs"
```

4.4. Incident Submission and Subsequent Redirection

The server side code at the bottom of the ASPX file controls the postback and redirection behaviors of the page. The submit button click event is registered in OnInit(). The click handler is implemented in Co3Submit_Click(). If the incident is created, the user is redirected to the URL specified in the click handler.

Notice that you can specify preset values in the click handler. This allows you to save user specific information with the incident without needing the user to provide it.

4.5. Available fields

You can only use controls from the System.Web.UI.WebControls namespace. Each field must use a control that supports the field's data type. The page displays an error if there is a data type mismatch.

Each field is identified uniquely by its designated ID as listed below. DropDownList, RadioButtonList and CheckBoxList controls are automatically populated with data specific to the fields they represent.

Field Name	Allowable ASP.net Types	Description
co3_addr	TextBox	Physical location of the incident, if applicable.
co3_hard_liability	TextBox	Assessed Liability
co3_city	TextBox	City
co3_country	DropDownList RadioButtonList	Country
co3_crimestatus_id	DropDownList RadioButtonList	Criminal Activity
co3_data_compromised	DropDownList RadioButtonList	Whether sensitive or personal data was foreseeably exposed and/or compromised. A value of "Yes" or "Unknown" indicates that a breach response may be required.
co3_data_encrypted	DropDownList RadioButtonList	Whether the data in question was encrypted.
co3_data_format	DropDownList RadioButtonList	Specifies the format of the personal information involved.
co3_discovered_date	TextBox	Date the incident was discovered/reported - this is the date upon which most reporting/action timelines are based, so it is important to ensure accuracy for this field.
co3_start_date	TextBox	Date the incident occurred.
co3_exposure_dept_id	DropDownList RadioButtonList	Department
co3_description	TextBox	Free form text description of the incident.
co3_employee_involved	DropDownList RadioButtonList	Employee Involved

Field Name	Allowable ASP.net Types	Description
co3_data_contained	DropDownList RadioButtonList	Whether the exposure has been addressed and rectified.
co3_exposure_type_id	DropDownList RadioButtonList	Origin source of the exposure
co3_harmstatus_id	DropDownList RadioButtonList	Harm Foreseeable
co3_confirmed	DropDownList RadioButtonList	Tag an issue as an unconfirmed (event) vs a confirmed incident.
co3_incident_type_ids	CheckBoxList	Type of incident
co3_exposure_individual_name	TextBox	Individual Name
co3_jurisdiction_name	TextBox	Jurisdiction
co3_name	TextBox	Unique name to identify this particular incident.
co3_negative_pr_likely	DropDownList RadioButtonList	If it is foreseeable that the incident might generate any negative public image or publicity for your company or organization.
co3_nist_attack_vectors	CheckBoxList	NIST Attack Vectors the incident falls under.
co3_phase_id	DropDownList RadioButtonList	Phase of the incident.
co3_postal_code	TextBox	Postal Code
co3_province	DropDownList RadioButtonList	Canadian provinces
co3_reporter	TextBox	Name of person who reported the event, such as a device owner or his/her manager.
co3_resolution_id	DropDownList RadioButtonList	Select an option that accurately describes the reason for closing this incident.
co3_resolution_summary	TextBox	Enter a summary that describes how this incident was resolved.
co3_severity_code	DropDownList RadioButtonList	Your impression of the events relative severity vs. other events that may be entered into the system.

Field Name	Allowable ASP.net Types	Description
co3_inc_training	DropDownList RadioButtonList	Whether the incident is a simulation or a regular incident. This field is read-only.
co3_data_source_ids	CheckBoxList	Original source of the data, such as the name of the database.
co3_state	DropDownList RadioButtonList	United States states
co3_plan_status	DropDownList RadioButtonList	Status
co3_exposure_vendor_id	DropDownList RadioButtonList	Vendor
co3_zip	TextBox	Zip (or postal) code of the location of the incident.

4.6. Custom Fields

Custom fields can be added using their corresponding System.Web.UI.WebControl. The control's ID is the API Access Name prefixed with "co3_". For example, if the API Access Name is "employee_id", then the control's ID is "co3_employee_id".

Custom Field Type	Allowable ASP.net Types
Date Picker	TextBox
Date Time Picker	TextBox
Text	TextBox
Number	TextBox
Text Area	TextBox
Select	DropDownList
RadioButtonList	Boolean
DropDownList	RadioButtonList
Multiselect	CheckBoxList

5. Troubleshooting

If an error occurs or an exception thrown, the Code-behind invokes the error handling function with the following signature:

```
protected override void OnHandleException(Exception ex, string message)
```

You can implement this method to display the error message. The ASPX file implements this method to provide a generic error message. You can expand this method to provide better error handling, or remove it.

End of document