# Security of DataFlex Web Framework Applications

Version: 1.4

## Overview

Reliable security is critical in web applications. In DataFlex 2021, security has been enhanced by adding automatic validations to security-related web properties to make web applications more secure without code changes. For applications using DataFlex revisions 18.2 through 19.1, we have created a new “Redaction Library” that allows developers to improve security themselves using techniques similar to the DataFlex 2021 implementation. We will review both sets of security improvements in this document.

## Introduction

The Web Framework provides application access control at the view level. By default, users need to log in before they can access anything within the application. This behavior can be configured using the peLoginMode (cWebApp) and pbLoginModeEnforced (cWebWindow) properties. Further customization can be done by implementing the AllowAccess (cWebWindow) and AllowViewAccess (cWebSessionManager) event hooks. When doing this, the framework will make sure that all access to the views is blocked at the lowest level.

It is also common practice in web apps to hide or disable UI elements within a view or a menu based on user rights. Doing this in 18.2 through 19.1 using pbRender / pbVisible or pbEnabled is not secure because these properties are client web properties, which means that they are implemented in JavaScript and their value is stored within the browser. Unfortunately, this allows the stored value to be changed using the browser’s developer tools. The capability to change stored values can be exploited by bad actors to access data they should not have access to or to execute functionality that should not be available to them.

Awareness of this risk is too low, which can lead to unintentional vulnerabilities in DataFlex web applications that use those techniques. For applications that are publicly accessible, this can be a serious vulnerability. While developers can program around this vulnerability, the 18.2 through 19.1 frameworks lack the hooks to do so easily. In DataFlex 2021, we enhanced security by adding automatic validations based on pbRender, pbVisible and pbEnabled to make applications more secure without requiring code changes. For applications using prior revisions (18.2 through 19.1) we have created a new “Redaction Library” that allows developers to improve security themselves. We will review both sets of improvements in this document.

### Vulnerability

The vulnerability can manifest in two main ways: through manipulating “hidden” data entry objects and triggering events on the server.

#### *Data Entry Objects*

When hiding a data entry object using pbRender, the control is hidden from JavaScript using CSS. If the browser’s development tools are used to unhide this control, it shows the current field value. In addition, changes made to this value will automatically be saved if the DD allows this (creating DD level protections is one of the techniques currently used by developers).

*Server Actions (Events)*
Even if a control is hidden or disabled, the server will accept server actions for it. For instance, if a button is hidden or disabled, it is still possible to trigger OnClick using the browser's developer tools.

## DataFlex 2021
The Web Framework controls in DataFlex 2021 are more secure by default through improvements to the pbRender, pbVisible and pbEnabled properties. This means that without code changes, DataFlex 2021 web applications are more secure.

### ClientProtected Web Properties
DataFlex 2021 supports a new web property type: ClientProtected. This web property type is sent to the client but its value is also stored on the server (like a Server web property). Performing a WebGet on these properties will always return the server value. Performing a WebSet will update both the server and the client-side value. The class library uses the ClientProtected properties to perform extra security checks.

### IsControlAccessible
The properties pbRender, pbVisible and pbEnabled for all controls are ClientProtected properties. This gives us a reliable value on the server to create additional checks in the framework classes. Because disabling or hiding a container will disable or hide all its children, just checking these properties for the current object is not enough. A new function, IsControlAccessible, checks if a control is visible and/or enabled. This function also looks at the state of parent objects and will check if the view in which the control is located is synchronized (when necessary). A caching system is built-in to make this call as efficient as possible (in some cases it will be called a lot).

### AllowServerAction (cWebObject)
Before a published function or procedure is called (from the client), AllowServerAction is called. This provides the basic hook for controlling server actions based on user rights. If the bAllow parameter is set to False, the call is not processed and an error is returned to the client. The cWebBaseUIObject class (the base class for all UI objects) overrides this procedure and checks if a control is accessible using IsControlAccessible. This means that only controls that are enabled and visible on the client will now process events.

This procedure can be overridden to customize the above described behavior. Note that not doing a forward send will make it skip the new security check. Some controls might override this procedure to enable specific server actions when a control is disabled.

### AllowFillDEO (cWebBaseDEO)
This procedure is called whenever a data-aware control loads a value from the Data Dictionary. For a cWebForm, this procedure is called when a record is found and the Refresh message is received from the Data Dictionary. For a cWebColumn, this procedure is called for every list row that is loaded. If bAllow is set to False, the control does not load the value. The default is that cWebBaseDEO calls IsControlAccessible to verify if the control is visible (not hidden).

Note that this does not check pbEnabled since disabled controls can still display values. The property **pbNoFillIfHidden** can be set to False to make a hidden DEO still receive a value. This is used when the control is hidden for a different reason than the user not being allowed to see the value.

Also note that when unhiding a data-entry object the object will not contain the right value. Send Request_Assign to the data dictionary to trigger an explicit update.

### AllowUpdateDD (cWebBaseDEO)

This procedure is called whenever a data-aware control updates the data dictionary with a changed value. This happens as part of the DDO synchronization that is done for each request. It will validate that the update is allowed and if not (bAllow is False), the Set Field_Changed_Value is not performed. By default, IsControlAccessible is called to verify if the control is visible and enabled. The property **pbNoUpdateIfHidden** can be set to allow hidden controls to update the DD.

### Setting the pbRender, pbEnabled and pbVisible Properties

To make the security checks function properly, it is important that WebSet is used when setting these properties based on the user rights. In addition, any time client web properties are altered in OnLoad using Set (to prevent them from becoming synchronized), these ClientProtected web properties must be WebSet. For example, this is the proper way to hide a control based on user rights:

```
Object oCustomerCredit_Limit is a cWebForm
    Entry_Item Customer.Credit_Limit
    Set piColumnSpan to 6
    Set piColumnIndex to 0
    Set psLabel to "Credit Limit:"
    Set peLabelAlign to alignRight
    Set pbRender to False

    Procedure OnLoad
        Integer iRights

        Get piUserRights of ghoWebSessionManager to iRights

        If (iRights > 1) Begin
            WebSet pbRender to True
        End
    End_Procedure
End_Object
```

For controls outside a view (like menu items) you must update this state after log in. The OnRightsChange event is available on menu items and the cWebApp class to do this:

```
Object oOrderListMenuItem is a cWebMenuItem
    Set psCaption to "Order List"
    Set pbRender to False

    WebRegisterPath ntNavigateBegin oOrderListSample
    Procedure OnClick
        Send NavigatePath
    End_Procedure

    Procedure OnChangeRights
        Integer iRights

        Get piUserRights of ghoWebSessionManager to iRights

        WebSet pbRender to (iRights = 1)
    End_Procedure
End_Object
```

Note that in both of these cases the default of pbRender is False and only for sessions that are allowed access it is WebSet to true. That provides the most secure environment.

When using the drilldown model a good place to WebSet these properties is inside OnNavigateForwardPreFindInit. This event is called earlier than OnNavigateForward itself, before the navigation finds are performed, so that the update of the DEO is allowed or denied according to the latest values.

## Redaction Library (DataFlex 19.1 / 19.0 / 18.2)

The Redaction Library allows developers to improve the security of their applications under existing DataFlex revisions. This library provides subclasses of all the standard DataFlex UI objects and extends them with the hooks to easily add checks to each server action and prevent DEO's from communicating with the Data Dictionaries. Since ClientProtected web properties are not available in earlier revisions, it was not possible to implement additional security through pbRender, pbVisbile and pbEnabled.

The subclasses are prefixed with R,  so cWebForm is cRWebForm. Developers can replace every control with the new subclass or replace just the ones where redaction is needed. The implementation does not require JavaScript changes. Use the AllRedectationWebAppClasses.pkg to include the subclasses in your project.

### All Web Objects

#### pbRedact (cRedaction_mixin)

This server web property is available for DataFlex 19.0 and 19.1 and can be set to True to indicate that the control is inaccessible. This event hooks will look at this property to allow access.

#### IsRedacted (cRedaction_mixin)

This function in 19.0 and 19.1 simply returns the value of pbRedact. It can be augmented to provide custom logic.In 18.2, due to the lack of server web properties, implementing this function is needed to enable the redaction.

#### AllowServerAction (cRedaction_mixin)

AllowServerAction is called before a published function or procedure is called (from the client). This provides the basic hook to prevent server actions from being executed based on user rights. If the bAllow parameter is set to False, the call will not be processed and an error is returned to the client. By default, it will call IsRedacted to determine if the call should be allowed. It can be augmented to customize this behavior.

### Data Entry Objects

#### pbNoFillIfRedacted (cRedactionDEO_mixin)

Controls if pbRedact will stop the DEO from filling itself with data (from data dictionaries). It defaults to true. Set this to false in case the control should still show data but shouldn't be able to save its data.

#### AllowFillDEO (cRedactionDEO_mixin)

AllowFillDEO is called whenever a data- aware control loads a value from the Data Dictionary. For a cWebForm, this procedure is called when a record is found and the Refresh message is received from the Data Dictionary. For a cWebColumn, this procedure is called for every list row that is loaded. If bAllow is False, the control will not load the value. By default, it calls IsRedacted.

### AllowUpdateDD (cRedactionDEO_mixin)

AllowUpdateDD is called whenever a data-aware is asked to update the data dictionary with a changed value. This happens as part of the DDO synchronization that is done for each request. Its purpose is to validate if the update is allowed and if bAllow to False the Set Field_Changed_Value is not performed. By default, it will call IsRedacted.

### Using Redaction in DataFlex 19.x

To implement redaction for a form in DataFlex 19.x, use the cRWebForm and set pbRedact to True (using WebSet) if the user should not have access to the data. The example below shows how to do this in OnLoad based on the user rights.

```
Object oCustomerCredit_Limit is a cRWebForm
    Entry_Item Customer.Credit_Limit
    Set psLabel to "Credit Limit:"
    Set peLabelPosition to lpTop
    Set pbRedact to True

    Procedure OnLoad
        Integer iRights

        Get piUserRights of ghoWebSessionManager to iRights

        WebSet pbRender to (iRights = 1)
        WebSet pbRedact to (iRights <> 1)
    End_Procedure
End_Object
```

The example below shows how to make sure a disabled button cannot be accessed.

```
Object oShowDialogBtn is a cRWebButton
    Set piColumnSpan to 0
    Set psCaption to "Settings"
    Set pbRedact to True
    Set pbEnabled to False

    Procedure OnClick
        Send Popup of oMyDialog
    End_Procedure
End_Object
```

### Using Redaction in DataFlex 18.2

To use redaction in DataFlex 18.2, implement IsRedacted (and use the cRWebForm). The example below shows how to do this based on the user rights.

```
Object oCustomerCredit_Limit is a cRWebForm
    Entry_Item Customer.Credit_Limit
    Set psLabel to "Credit Limit:"

    Procedure OnLoad
        Integer iRights

        Get piUserRights of ghoWebSessionManager to iRights

        WebSet pbRender to (iRights = 1)
    End_Procedure

    Function IsRedacted Returns Boolean
        Integer iRights
        Get piUserRights of ghoWebSessionManager to iRights

        Function_Return (iRights <> 1)
    End_Function
End_Object
```