# SUCCESS TIP: PROTECTING NETSUITE CUSTOM CODE

**How to protect your workflow customizations from unexpected errors**


## In this Success Tip:

In this success tip, we discuss the power of the NetSuite customization capabilities using industry-standard JavaScript.

We also look at how adding "wrapper" APIs around the existing NetSuite APIs can add value while helping manage unexpected errors and improve the ability to maintain your custom code over major NetSuite upgrades.

## Related Services:

Centricity Systems offers the following NetSuite related services for clients implementing or customizing their NetSuite solution.

- NetSuite JumpStart Service
- NetSuite Consulting Services

For more information, contact us at
1-603-589-8473
or email us at
csinfo@centricitysystems.com

NetSuite offers a fantastic environment using industry standard JavaScript that allows you to customize and manage workflow and form/data entry so that you can implement very specific workflow processes for various events (e.g. opportunities, orders, cases, leads, etc). However, one of the most frustrating things that can happen is when one of NetSuite's JavaScript-based functions (or API) does not return the expected result or performs an unexpected behavior due to an inadvertent update or maintenance release by NetSuite.

These things can happen. It is not because of bad or unreliable code, but when you are continually updating an application as highly complex as NetSuite, well, these things do happen. Fortunately, NetSuite is pretty fast at fixing these events when they occur. However, during the period between the error and correction, your code may be broken and could potentially prevent you from performing your job or from properly setting values in a workflow/data entry sequence.

In version 2009.x, NetSuite has introduced better error handling that now provides a popup alert to the user when uncaught errors occur. Even though this is tremendous help for troubleshooting purposes, it does not necessarily allow the specific function you may have called to gracefully manage the error, thus, your workflow process may not execute as you expect it to.

## The Value of "Wrapper" APIs

To protect against unexpected behavior, and to add your own layer of management to the underlying NetSuite code, it is a good idea to implement a "wrapper" library as part of your overall framework. This library provides functions or custom APIs that literally "wrap" around the NetSuite function/API and handle any errors the way you need them to, or add additional functionality so that your workflow and scripts can continue or adjust accordingly should there be an unexpected error.

To understand the value of a "wrapper" API, the following workflow function is used as an example.

```
function myWorkflow() {
  // get the value from the specified field ID and
  // remove any spaces or non-alphanumeric characters
  var val = nlapiGetFieldValue('fieldid');
  if ( val == null || val == '' ) { return; }
  // run regex
  var new_val = val.replace(/[^A-Za-z0-9_]*/g,'');
  if ( val == new_val ) { return; }
  // reset the field to the new value; don't fire event
  nlapiSetFieldValue('fieldid',new_val,false);
} // end workflow/entry validation process
```

The above workflow example is designed to ensure that a specific field (identified by 'fieldid') only contains alphanumeric characters (no spaces, periods, commas, etc) or is empty. However, if there is an error internally within the NetSuite nlapiGetFieldValue API, it is possible that the error will cause the program to stop executing at this line and return or blow out of the 'myWorkflow' function without ever completing the lines of code below it. Thus, it would be possible for an invalid set of characters to be entered and saved in the form (something you do not want), causing more extensive problems.

One way to prevent code "blow-out" and ensure code execution would be to wrap the code inside the 'myWorkflow' function in a try/catch to catch any errors and manage them. This of course, is good practice in general, however, not very practical simply to manage unexpected errors in a NetSuite API that is potentially used in tens of hundreds of places throughout your many custom scripts.

## A Better Mousetrap

A better solution is to create a library of wrappers that you call instead of the NetSuite APIs. Within these wrapper functions, you can manage errors, provide additional functionality, and improve code maintenance. In addition, wrappers help alleviate the issues with "API changes" where a vendor may decide due to needed upgrades to the product, to change the behavior of an existing API, or deprecate it for one or more newer APIs.

Vendor changes such as these can cause major headaches in terms of backwards compatibility of existing code and can force you to go back through all your existing code to update any calls to this changed/new API (I'd like to say this doesn't happen but unfortunately, it does — and not just with NetSuite).

Using a wrapper, you would only need to make this change in one place and all existing code would continue to function as it always has.

Another benefit to using wrapper APIs is the ability to add functionality that the built-in APIs do not offer. For example, the NetSuite API nlapiGetFieldValue used in the above example, performs a call that returns the value in the specified field ID. If the field does not exist, null is returned.

One such added functionality incorporated into our NetSuite API wrapper is a default value returned when null or an error occurs. Using a wrapper, you can add this functionality. Thus, the following API wrapper can be constructed.

```
// API "wrapper"
function csapiGetFieldValue(fieldid) {
  // param 2 = (optional) default val to return, otherwise null
  var dflt = (arguments.length>1?arguments[1]:null);
  // param 3 = (optional) alert on error. Default is false
  var pop=(arguments.length>2&&arguments[2]===true?true:false);
  var val = dflt;  // initialize return value to default
  try {
     val = nlapiGetFieldValue('fieldid');
     if ( val == null ) { val = dflt; }
  } catch(e) {
     // optional pop up an alert if error
     if ( pop === true ) { alert([popup message to user]); }
     val = dflt;  // reset value to the default return value
  }
  return val;  // return the field value or default value
} // end csapiGetFieldValue wrapper API
```

Now using the new API wrapper, we can rewrite our workflow function to handle unexpected errors and return a default value on error or null.

```
// new workflow handling
function myWorkflow() {
  // get value and remove non-alphanumeric characters.  Return
  // Boolean false on error and pop user alert.
  var val = csapiGetFieldValue('fieldid',false,true);
  if ( val === false ) {
     // unknown error, do what you need to do (e.g. cancel,
     // perform a fallback, etc)
     return false;
  }
  // continue rest of workflow code as usual
  ...
} // end workflow/entry validation process
```

Using the new wrapper API, the code can be managed much easier, can handle unexpected errors without potentially crippling the scripts execution, and adds additional features specific to your needs without compromising the internal NetSuite API.

Additionally, any changes or updates can be performed in one single place making it much easier to manage customizations through the major upgrades that occur twice a year in NetSuite.

## NetSuite Implementation Services

Centricity Systems has been a NetSuite implementation provider for over 3 years, helping clients to implement, customize, and take full advantage of the powerful NetSuite platform.

Centricity Systems offers NetSuite Implementation Services to help customers quickly and successfully plan, deploy, train, and customize/integrate their NetSuite solution.

For more information on our NetSuite Implementation Services, visit us at www.centricitysystems.com or call us at +1.603.589.8473 for a no-obligation discussion on how we can help you with your NetSuite implementation.

### Contact Us

To discuss how we can help your organization, call us today at 1 (603) 589-8473 or e-mail us at:

csinfo@centricitysystems.com

Visit us on the web:

www.centricitysystems.com

### About Centricity Systems

Centricity Systems is an independent consulting firm specializing in helping clients in the effective use of Process and Information technology to solve complex problems, improve operations, and implement streamlined efficiencies in a cost-conscious manner.

The company was founded in 2002 from a long and successful list of experiences spanning 25 years helping companies define, implement, and manage solutions and services that drive business efficiency, high-performance, and shareholder value.

Centricity Systems offers a unique combination of deep industry expertise and broad business operations know-how to offer unparalleled experience and perspective in executing and delivering the highest value and return for clients on every engagement.

The fact that clients continually re-engage Centricity Systems for further services is testament to the quality and value that is always delivered.