Populate template Client Script Documentation

Objective

When a new Incident is created by a user on the platform, they fill in the caller's information using the caller's email address. This automatically populates the Location field on the form. Depending on the kind of issue the caller is facing, a template is selected (Internet issue, manager's pc etc), template that will populate the short description and the description, in order to guide the user to fill in the correct information for the incident to be handled.
By filling up the caller's email address in the first part of the newly created incident, the user already populates or retrieves some of the information that will be needed in the short description and the description (Email address, Store name and number), while the rest of the information necessary in the first part of the template is accessible by a call to the server (Contact phone number, Store full address). But the user has to type or copy/paste the information in the template itself. This is a waste of time, a task that can get tedious when the volume of tickets grows, and is error-prone.
By automating this process, it is possible to reduce errors, and make the task easier for the user, this is the objective of this script.

Dependencies

The caller's email address has to be filled in, it has to exist in the Users table and the user's record must have a reference field to a valid location, a store in this case, which would have a valid Address and store number. The Location field must appear on the incident form, if it does not, add it in Form Design.

Script type

This client script is an onChange script, created for the Incident table, focusing on the Description field, to monitor the application of a template.

Steps to create the client script:

a. Log in to your ServiceNow instance with appropriate access rights.
b. Navigate to System Definition > Client Scripts.
c. Click the "New" button to create a new client script.
d. In the "Name" field, provide a descriptive name for the script.
e. In the "Table" field, select the table where the form is located.
f. In the "Type" field, choose "onChange" as the script type.
g. In the "Field name" field, choose the email address field that triggers the script when changed.
h. In the "Script" field, write the JavaScript code for the client script.

Script code

```
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading || newValue === '') {
        return;
    }

    if (isTemplate == true) {
        try {
            // Get the sys_id of the caller
            var caller = g_form.getValue('caller_id');
            // Server call to retrieve email with sys_id
            var callerInfo = new GlideRecord('sys_user');
            callerInfo.get(caller);
            // Caller Email address
            var callerEmail = callerInfo.email;

            // Get sys_id of the location
            var location = g_form.getValue('location');
            // Server call to retrieve store info with sys_id
            var storeLocation = new GlideRecord('cmn_location');
            storeLocation.get(location);
            // Get store name
            var storeName = storeLocation.name;
            // Get store number
```

```
            var storeNumber = storeLocation.u_store_number;
            // Get store street
            var storeStreet = storeLocation.street;

            // Get the short description
            var shortDesc = g_form.getValue('short_description');
            // Split into array with /
            var splitShortDesc = shortDesc.split("/");
            // Update relevant entries of newly created array
            var updatedShortDesc =
splitShortDesc.map(function(lineOfShortDesc) {
                if (lineOfShortDesc === 'StoreNumber ') {
                    return storeNumber + ' ';
                } else if (lineOfShortDesc === ' City ') {
                    return ' ' + storeName + ' ';
                } else {
                    return lineOfShortDesc;
                }
            });
            // Join array into string
            shortDesc = updatedShortDesc.join("/");
            // Set short desc value with updated info
            g_form.setValue("short_description", shortDesc);

            // Get the description
            var desc = g_form.getValue('description');
            // Split into array with \n
            var splitDesc = desc.split("\n");

            // Update relevant entries of newly created array,
anticipating possible lines according to the different applicable
templates
            var updatedDesc = splitDesc.map(function(lineOfDesc) {
                if (lineOfDesc === 'Contact Phone Number: ') {
                    return lineOfDesc + ' 70' + storeNumber + 'xx';
                } else if (lineOfDesc === 'Store Full Address: ') {
                    return lineOfDesc + storeStreet;
                } else if (lineOfDesc === 'Contact e-mail: ') {
                    return lineOfDesc + callerEmail;
                } else if (lineOfDesc === 'Store Name & Number: ') {
                    return lineOfDesc + storeName + ' ' + storeNumber;
                } else {
                    return lineOfDesc;
```

```
                }
            });
            // Join array into string
            desc = updatedDesc.join("\n");
            // Set desc value with updated info
            g_form.setValue("description", desc);
        } catch (err) {
            // Catch any errors that occur and log them to the console
            console.log('An error occurred: ' + err);
        }
    }
}
```

Testing

This script has been tested in these 4 different situations all followed by the application of a template when possible:

Test conditions:

a. Valid email addresses associated with existing users and locations.
b. Email addresses not associated with any user.
c. Email addresses associated with users but not locations.
d. Empty or invalid email addresses.

Results:

a. The template is populated with the relevant information, as expected.
b. The message “Invalid reference” appears under the caller reference field, as expected.
c. The template is populated with the caller’s information, but the location/store’s lines of the template are undefined, as expected.
d. The message “Invalid reference” appears under the caller reference field, as expected.

The tests have been performed using Google Chrome, Mozilla Firefox and Microsoft Edge.

Deployment

In order to ensure a smooth transition, this script must follow a structured deployment process that includes moving the script through development, test, and production instances using update sets. It will have to be tested in the new environments using the same steps as described before.