Custom On-Device Validation

This feature allows you to override certain app-side action outcomes with JavaScript executed locally on-device.

To configure this to your Service use the API variable `custom_validation_script` or go to the **Advanced** when editing your service on the website and paste your code under option `Enable on-device custom validation.`

Validation Functions

These are the functions you can implement to override normal processing.

- `onSessionSave` - Executed when app-users save answers to `Shared Questions` under `Session Info`.
- `onCreateScan` - Executed when app-users take action to create new Scan records.
- `onUpdateScan` - Executed when app-users take action to update existing Scan records.

Input "data" Parameter

The input parameter of these validation functions is a JSON object with the pertaining information about the Scan record at the time of the given action. Some fields are not present for some actions because the data doesn't exist yet.

Full "data" Example


```
{
   "scanValue":"050428375624",
   "scanStatus":"1",
   "scanResponse":"Recorded to device.",
   "answers":[
      {
         "qtype":"manual",
         "value":[ "Donated $5 to the cause." ],
         "qtext":"Comments:",
         "eid":"",
         "qid":"1525"
      },
      {
         "qtype":"checkbox",
```

```
            "value":[ "yellow", "green" ],
            "qtext":"Colors",
            "eid":"",
            "qid":"303"
         }
      ],
      "properties":{
         "time_zone":"GMT-04:00, GMT-4",
         "capture_type":"camera_scan",
         "app_used":"2.11.14r1625 for iOS-10.2.1",
         "barcode_format":"UPC_A",
         "scanned_at_utc":"2017-06-02T23:32:00.260Z",
         "scan_engine":"qmv1.9.6r4502"
      },
      "context":{
         "username":"MyUser",
         "userId":"39430",
         "deviceId":"39434",
         "serviceId":"9859",
         "serviceName":"MyService",
         "isCloudScan":false
      }
   }
```


`onSessionSave(data)`

- `data` param only includes `answers`, `properties`, and `context`.
- Return `JSON.stringify({})` to continue as normal.
- Return `JSON.stringify({"errorMessage":"MESSAGE"})` to stop the action from completing.

`onCreateScan(data)`

- `data` param does not contain `scanStatus` and `scanResponse`.
- Return `JSON.stringify(data)` (the unchanged input param) to continue as normal.
- Return `JSON.stringify({"errorMessage":"MESSAGE"})` to stop the action from completing.
- Return an altered response: `JSON.stringify({"scanValue":"OVERRIDE","scanStatus":"OVERRIDE", "scanResponse":"OVERRIDE"})`

- Note: when `data.context.isCloudScan == true`, you cannot alter `scanStatus` and `scanResponse` because those values will be determined server-side.

`onUpdateScan(data)`

- Return `JSON.stringify(data)` (the unchanged input param) to continue as normal.
- Return `JSON.stringify({"errorMessage":"MESSAGE"})` to stop the action from completing.
- Currently we do not allow this script to alter `scanValue`, `scanStatus` or `scanResponse` when updating a previously created Scan record.

Internal Util Functions

These functions are available to aid in more complex validation cases.

`window.CRHOOK.putString(key, value)`

Saves string param `value` under the string param `key` to enable validation logic that depends on persistence. This key/value will only be stored locally and accessible to the current device and user.

`window.CRHOOK.getString(key, defaultValue)`

Returns the value stored with `putString(key, value)` or string param `defaultValue` if none exists.

`window.CRHOOK.putSharedString(key, value)`

Same as `putString(key, value)` except value will be accessible to the given users across all their services instead of just the current.

`window.CRHOOK.getSharedString(key, defaultValue)`

Returns the value stored with `putSharedString(key, value)` or string param `defaultValue` if none exists.

`window.CRHOOK.dbSearch(value, exactValue, response, exactResponse, validity, limit, isOrSearch)`

Search the locally download item database of the currently selected Service if it exists. Pass null to any value you want to exclude from the filter.

- `@param value` Filter for the primary value i.e. barcode
- `@param exactValue` When `true` the value filter must be an exact match.
- `@param response` Filter for the value's response text i.e. description, details, etc.
- `@param exactResponse` When `true` the response filter must be an exact match.
- `@param validity` Set to `"0"` or `"1"` to match value's validity or `null` to not filter at all. This is not a boolean param.
- `@param limit` Maximum number of results to return.
- `@param isOrSearch` Set to `true` to do an *OR* search instead of an *AND* search when both value and response filters are supplied.
- `@return` list of objects from the database in the format below or empty list when matches are found.


```
[
   {
      "value":"BARCODE_01",
      "response":"SOME RESPONSE",
      "isValid":"1",
      "scanCount":"0"
   },
   {
      "value":"BARCODE_02",
      "response":"Another response",
      "isValid":"1",
      "scanCount":"3"
   },
   {
      "value":"BARCODE_n",
      "response":"",
      "isValid":"0",
      "scanCount":"0"
   }
]
```


Code Example

This custom validation demonstrates blocking the action with an error message and altering the scan value.

```
<script>

let qidSession = "2048086";
let qidUpdate = "2026859";
let startDate = "2021-06-15T21:00:00Z";

function onSessionSave(data) {
    let answer = getAnswer(data, qidSession);
    if (answer == null) {
        return JSON.stringify( {"errorMessage": "Question["+qidSession+"] not found!"} );
    }
    let ansVal = answer.value[0];
    let searchResults = window.CRHOOK.dbSearch(ansVal, true, null, false, null, 1, true);
    let results = JSON.parse(searchResults);
    if (results.length == 0) {
        return JSON.stringify( {"errorMessage": "Answer must be in the database.\nFix Question:\n\""+answer.qtext+"\""}
);
    }
    return JSON.stringify(data);
}

function onUpdateScan(data) {
    let answer = getAnswer(data, qidUpdate);
    if (answer == null) {
        return JSON.stringify( {"errorMessage": "Question["+qidUpdate+"] not found!"} );
    }
    let ansVal = answer.value[0];
    var prev = window.CRHOOK.getString("prev", "");
    if (ansVal.length > 0 && ansVal == prev) {
        return JSON.stringify( {"errorMessage": "Cannot answer same as last time.\nFix
Question:\n\""+answer.qtext+"\""} );
    }
    window.CRHOOK.putString("prev", ansVal);
    return JSON.stringify(data);
}

function onCreateScan(data) {
    var now = new Date();
    var valid_from = new Date(startDate);
    if (valid_from>now) {
```

```
        let result = {
            "scanValue": "early-" + data.scanValue,
            "scanStatus": "0",
            "scanResponse": "early-" + data.scanResponse
        };
        return JSON.stringify(result);
    }
    return JSON.stringify(data);
}

function getAnswer(data, qid) {
    if (Array.isArray(data.answers)) {
        for (var i=0; i<data.answers.length; i++) {
            let ans = data.answers[i];
            if (ans.qid == qid) {
                return ans;
            }
        }
    }
    return null;
}

</script>
```