Validate the combination of the primary scan value and a question-answer scan value or manual entry against a database.

Example: Scan product UPC then scan or enter the BIN number to place it in.

UPC code must be in the database
The BIN number must be in the database combined with the UPC code.
If the UPC code and BIN number are combined in the database, the app user can place the item into the BIN and then collect additional information (photo, text, location, signature, etc.).

Column 1 of the database would look like this if the COV is verifying that both the primary value and question-answer value are combined in the database:

UPC-BIN

UPC-BIN

UPC-BIN

Copy and paste the full script below into the Advanced step's form field for "Enable on-device custom validation."

<script>

```
// _valueA is required. _valueB and _valueC are option.
// Set to empty string i.e. "" if not needed.
// _valueA/B/C can be set to:
// "SCAN" - Represents the primary scan value
// "QUESTION" - Represents first question asked regardless of ID
// "1234" - Represents a specific question by ID
var _valueA = "SCAN";
var _valueB = "QUESTION";
var _valueC = "";
// The ability to combine A/B/C in whatever order you need to form
// the string that the db will be validated against with.
var _comboValue = "${a}${b}";
// Map regular expressions to alter _valueA/B/C if needed
var regex = {
    "a": [new RegExp("^[\\w]{3}([\\w]{7})[\\w]*$"), "$1"],
    "b": [new RegExp("^[\\w]{4}([\\w]{4})[\\w]*$"), "$1"],
};
// Set to true to also alter the scan value to the constructed _comboValue
```

```
var _alterScanValue = false;
// The message shown if _comboValue is not found in the database
const _msgInvalidScan = "Invalid. Scanned value not found in database.";
// Set to false to ignore duplicates or set to a message make them invalid.
const _msgInvalidDuplicate = false;
// Block submit with message if batch is considered invalid
const _blockSubmit = false;

////////////////////////////////

function onCreateScan(data) {
    let comboVal = getComboValue(data);
    let dbValue = validateValue(comboVal);
    if (dbValue != undefined) {
        if (_alterScanValue) data.scanValue = comboVal;
        if (dbValue.isValid == "1" && _msgInvalidDuplicate && dbValue.scanCount != 0) {
            data.scanResponse = _msgInvalidDuplicate+"\n\n"+dbValue.response;
            data.scanStatus = "0";
        } else {
            data.scanResponse = dbValue.response;
            data.scanStatus = dbValue.isValid;
        }
    } else {
        data.scanResponse = _msgInvalidScan;
        data.scanStatus = "0";
    }
    if (_blockSubmit && data.scanStatus == "0") {
        return JSON.stringify({"errorMessage":data.scanResponse});
    }
    return JSON.stringify(data);
}

function evalValue(data, valueId, regex) {
    var val = (valueId.toUpperCase() == "SCAN") ? data.scanValue :
getAnswerVal(valueId=="QUESTION"?false:valueId, data.answers);
    if (Array.isArray(regex)) {
        val = val.replace(regex[0],regex[1]);
    }
    return val;
}

function getComboValue(data) {
    var valA = evalValue(data, _valueA, regex.a);
    var valB = evalValue(data, _valueB, regex.b);
```

```
    var valC = evalValue(data, _valueC, regex.c);
    return _comboValue.replace("${a}", valA)
    .replace("${b}", valB).replace("${c}", valC);
}

function getAnswerVal(qid, answerArray) {
    if (Array.isArray(answerArray)) {
        if ( qid === false ) {
            if (answerArray.length > 0) {
                return answerArray[0].value[0];
            }
        } else {
            var ans = answerArray.find(a => a.qid == qid);
            if (ans) {
                return ans.value[0];
            }
        }
    }
    return "";
}

function validateValue(value) {
    let searchResults = window.CRHOOK.dbSearch(value, true, null, false, null, 1, true);
    let results = JSON.parse(searchResults);
    if (Array.isArray(results) && results.length > 0) {
        return results[0];
    }
    return undefined;
}

</script>
```