Validate a secondary scan value or manually entered value against a database.

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

UPC code is in the database. This is optional.
The BIN number must be in the database.

If the BIN number is in the database, the app user can place the item into the BIN and then collect additional information (photo, text, location, signature, etc.).

The database would optionally have the UPC code in column 1 of the database but must have the BIN in the database in column 1. The database would look like this if the COV is verifying that both the primary value and question-answer value are in the database:

UPC

UPC

UPC …

BIN

BIN

BIN …

The database would look like this if the COV is validating just the question-answer value:

BIN

BIN

BIN …

```
<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 = "QUESTION";
var _valueB = "";
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}";
// Map regular expressions to alter _valueA/B/C if needed
var regex = {
};
// 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 = true;
// Just return original response if not initially valid.
const _requireValidScan = true;

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

function onCreateScan(data) {
   if (_requireValidScan && data.scanStatus != "1") return JSON.stringify(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;//"searchResults of "+value+" = "+searchResults;//undefined;
}

</script>
```