

Secure Code Model

If there is an error, it should be found

Nothing dangerous should have happend

The user should be informed

The error information should be possible to log/save and the call stack and in-parameter to each script in the call chain should be possible to be examined.

Script response

Two questions:

- Did you do what your responsibility was?
- If not, tell me what happend
- If you did, give me the result (if it that is what you should do)

Script response

```
{ "status": ,  
  "result":  
}
```

Script response

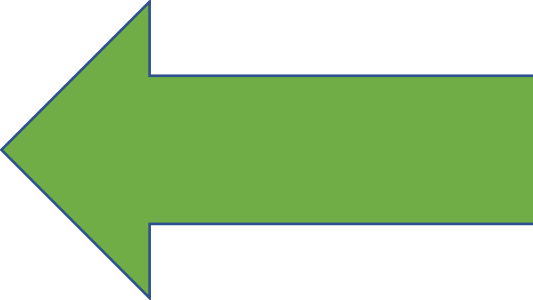
```
{ "status": "OK",  
  "result":  
}
```



The script has done
what it supposed to do!

Script response

```
{ "status": "OK",  
  "result": "": {  
    "message":  
    "Records created OK" }  
}
```



The **result** of the script is placed in a result-object.

Mandatory key in the result object is the key "message"

```
}
```

Script response

```
{ "status": "Error",  
  "result": {  
    "message": NIL  
  }  
}
```



The script **FAILED** to perform its mission for some reason.

Script response

```
{ "status": "OK",  
  "result": {  
    "message": "Records created OK",  
    "recordscreated": 42,  
    "duration_ms": 71  
  }  
}
```

Script response

```
{ "status": "Error",  
  "result": {  
    "message" : "An error has occurred in the system.",  
    "error": {  
      "scriptname": "myFaultyScript",  
      "errornumber": 0,  
      "callstack": [  
        {  
          "scriptname": "myFaultyScript",  
          "environment": "client",  
          "scriptparameter": {}  
        }  
      ]  
    }  
  }  
}
```



```
1 # Information of custom functions (CF) in use for Secure Code Model (SCM)
2 # (Updated 2020-05-03 MB)
3
4
5 # The Secure Code Model is based on a JSON structure to carry information of the response of a script,
6 # and most importantly, if an error is found in a script, to store error information.
7
8 # The json returned from a script contains two keys: "status" and "result".
9
10 # The "status" key can have two values: "OK" or "Error":
11 #     "OK" (the script could do it's task without error)
12 #     "Error" (the could not do it's task)
13
14 # The "result" key value is an json object, the "result object".
15 # A mandatory key in the result object is "message". The value of "message" is of type string, and can be empty,
16 # If the system key is OK, message can contain information to the calling script.
17 # If the system key is Error, message holds the error message.
18
19 # The other content of the result object, is dependent if the status key is OK or Error.
20 # If the status is OK, the result object has the keys and other objects described by the called scripts "api"-documentation.
21 # If the status key has the value Error, the result object contains error information. The error information must follow a defined structure.
22
23 # To produce the response object from the script, a library of customs functions is for your use.
24 # If status is OK the Script Response library is used to create the response object and the result keys and values.
25
26 # To read the respond object content in the caller script, the Script Result library is used.
27 # (You can of course get a specific key value from the response object by use of "result.keyname" by yourself if you want).
28
29 # If you need to return an error, you can use the Error-group in the Script Response library to return a valid response object with error information.
30
31 # When checking the result from a called subscript, there is also an Error-group in the Script Result library to use.
32
33
```

```
33 # 1.0 ----- SCRIPT RESPONSE -----
34
35
36 # The response object to be used as a parameter to Exit Script, has to be created.
37 # The created object is used as a parameter in the functions to add keys and values.
38
39 # 1.1 - OK response
40 # ScriptResponse_OK_Create - Creates the response json object with a status = OK
41 Set Variable [$scriptResponse; Value: ScriptResponse_OK_Create]
42
43 # To add a value to the mandatory key "message", a specific function exist. It is optional to use this function.
44 Set Variable [$scriptResponse; Value: ScriptResponse_OK_AddResultMessageKeyValue ( $scriptResponse ; $message )]
45
46 # ScriptResponse_OK_AddResultString, ScriptResponse_OK_AddResultNumber and ScriptResponse_OK_AddResultBoolean - Adds key/value to the result object in the response object
47 Set Variable [$scriptResponse; Value: ScriptResponse_OK_AddResultString ( $scriptResponse ; $key ; $value )]
48 Set Variable [$scriptResponse; Value: ScriptResponse_OK_AddResultNumber ( $scriptResponse ; $key ; $value )]
49 Set Variable [$scriptResponse; Value: ScriptResponse_OK_AddResultBoolean ( $scriptResponse ; $key ; $value )]
50
51 # - Add a json object to the result object, an example
52 # If a more complex value, as an json object or an json array, is to be included in the result, this can be stored in the respond object in the result key.
53 Set Variable [$complexObject; Value: "{}"]
54 Set Variable [$complexObject; Value: JSONSetElement ( $complexObject ; "demoDate" ; "2020-03-31"; JSONString )]
55 Set Variable [$complexObject; Value: JSONSetElement ( $complexObject ; "demoTitle" ; "Secure Code Model demo"; JSONString )]
56 # Add object to the result key in the respond object
57 Set Variable [$scriptResponse; Value: JSONSetElement ( $scriptResponse ; "result" ; $complexObject ; JSONObject )]
58
```

```

59 # 1.2 Error response
60 # 1.2.1 - Error response for initial error discovery
61 # ScriptResponse_Error is used as general function to respond with an error in the current script.
62 # Example:
63 Exit Script [Text Result:ScriptResponse_Error ( $errorMessage ; $errorCodeNumber ; $inputParameter )]
64 # where the $inputParameter should contain the script parameter to the current script.
65
66
67 # 1.2.2 - Error response when status key is "Error" in sscript result from called subscript
68 #
69 # The most generic model is to test for "erronous" result from the called script and to pass the error up in call chain.
70 Exit Script [Text Result:ScriptResponse_Erroneous ( $scriptResultObj ; $inputParameter )]
71 # where $scriptResultObj is the subscript result, and $inputParameter is the script parameter to the current script
72
73 # The check of the result from the subscript be more detail, if the result is valid as such or if it is an error in a valid error response.
74 # - If result is inValid:
75 Exit Script [Text Result:ScriptResponse_ErrorInvalid ( $scriptResultObj ; $inputParameter )]
76 # - If result is valid and status is Error:
77 Exit Script [Text Result:ScriptResponse_ErrorPass ( $scriptResultObj ; $inputParameter )]
78 # In the second case, the error is passed up in the script chain, since it is a valid script result.
79
80 # 1.2.3 Error check in "single pass loop"
81 # Another tool used in check for error, is to use a specific library of custom functions, ErrorCheck_SingleLoop.
82 # This library use the "If-block-model" to enclose the single pass loop.
83 # Minimal example:
84
85 If [ErrorCheck_SingleLoopBlock // Initiates private variables and returns True to create a "code block"]
86     Loop
87         Exit Loop If [ErrorCheck_SingleLoopExit ( $logical_test_1 ; "error message 1")]
88
89         Exit Loop If [ErrorCheck_SingleLoopExit ( $logical_test_2 ; "error message 2")]
90         # etc.
91
92         Exit Loop If [True // Mandatory step to avoid a "endless loop"]
93     End Loop
94     If [ErrorCheck_SingleLoopErrorExist]
95         Set Variable [$errorMessage; Value: ErrorCheck_SingleLoopErrorMessage]
96
97         # If this is a check in a subscript in a script chain and the error will make the script to exit:
98         Exit Script [Text Result:ScriptResponse_Error ( ErrorCheck_SingleLoopErrorMessage ; 0 ; $inputParameter )]
99     End If
100 End If
101 # ErrorCheck_SingleLoopBlock          Need to be used, especially if multiple errorCheck_SingleLoop is used in the same script.
102 # ErrorCheck_SingleLoopExit          Returns true and set private variables.
103 # ErrorCheck_SingleLoopErrorExist    Returns true if there was an true logical test in a ErrorCheck_SingleLoopExit parameter.
104 # ErrorCheck_SingleLoopErrorMessage Returns the error message for the first ErrorCheck_SingleLoopExit with a true logical test.
105

```

```
108 # 2.0 ----- SCRIPT RESULT -----
109 # After calling a subscript that use the Secure Code Model (SCM), the script result should be a json object with structure as defined in :
110 # A common method is to start to check if the result is erroneous ( not valid or status is Error).
111
112 # 2.1 Check if script result from subscript is erroneous
113 If [ScriptResult_isErroneous ( $scriptResultObj )]
114     Exit Script [Text Result:ScriptResponse_Erroneous ( $scriptResultObj ; $inputParameter )]
115 End If
116 # This function check both for invalid script result as well as if the script result is valid but with status = Error.
117
118
119 # 2.2 Check if script result from subscript is invalid or has status = Error
120 # Check first for validity with the function ScriptResult_isInvalid
121 # If valid sccript result, check for status = error with ScriptResult_isError.
122 # Example:
123 If [ScriptResult_isInvalid ( $scriptResultObj )]
124     Exit Script [Text Result:ScriptResponse_ErrorInvalid ( $scriptResultObj ; $inputParameter )]
125 Else If [ScriptResult_isError ( $scriptResultObj )]
126     Exit Script [Text Result:ScriptResponse_ErrorPass ( $scriptResultObj ; $inputParameter )]
127 End If
128
129 # 2.2.1 Information from the error object
130 # To get the error message, you can use:
131 Set Variable [$errorMessage; Value: ScriptResult_message ( $ScriptResultObj )]
132
133 # To get the name of the script where the error happend, you can use:
134 Set Variable [$errorScriptName; Value: ScriptResult_valueGet ( $ScriptResultObj ; "scriptname" )]
135
136 # To get the error number where the error happend, you can use:
137 Set Variable [$errorNumber; Value: ScriptResult_valueGet ( $ScriptResultObj ; "errornumber" )]
138
139 # To get the list of the call stack from the current script to the error script, you can use:
140 Set Variable [$ScriptCallStack; Value: ScriptResult_ErrorCallStack_ScriptNamesList ( $ScriptResultObj )]
141
142
143 # 2.3 Status = OK, no error from subscript
144 # The function ScriptResult_isOK can be used to check if we have a status = OK. Example:
145 Set Variable [$check; Value: ScriptResult_isOK ( $scriptResultObj )]
146 # This is normally not needed, since the error check is done before.
147
148 # To make it easy to get the result values from keys, a CF to retrieve the values from the result object exist: ScriptResult_valueGet
149 # Example:
150 Set Variable [$valueOfKeyName; Value: ScriptResult_valueGet ( $ScriptResultObj ; "keyName" )]
151
152 # For the mandatory key "message" a specific function can be used, ScriptResult_message.
153 # Example:
154 Set Variable [$message; Value: ScriptResult_message ( $ScriptResultObj )]
```