Proprietary Notice

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

1.1.1 ProDAS (Professional Data Acquisition System) is a data acquisition system for gas turbine test cells. This specification defines the technical requirements for the interface offered by the Recalculation GUI Server.

1.1.2 The Recalculation GUI Server provides access to steady state data from the Test Result and Sensor Calibration Database and allows repeating the calculations that have been executed during the test run (i.e. in real time) with modified configuration data.

1.1.3 Clients of the Recalculation GUI Server are restricted to the recalculation procedures. The recalculation procedures shall be started only from the Recalculation GUI. The clients of the Recalculation GUI Server will enable users to control recalculation processes.

1.1.4 This document defines the COM interface to be used by the clients.

1.1.5 This document contains a number of Visual Basic Script examples illustrating the interfaces. Although these examples are tested working scripts, they are by no means complete Visual Basic Scripts but concentrate on the functionality of the Recalculation GUI. For instance, a VBS programmer would be well advised to enforce explicit declaration of variables.

1.2 Scope

1.2.1 This document is intended for programmers of the COM client components using the COM interface(s) specified herein as well as the programmers of the server program offering the COM interface(s).

1.3 Applicable Documents

Number	Title
ES78001.2620	Functional Requirements Document for proDAS
ES78023.2672	Engineering Specification for Recalculation GUI
DB78023.2852	Design Brief for Recalculation GUI
ICD78031.2661	Interface Control Document for Configuration Server
ICD78031.2664	Interface Control Document for Windows Database Server

1.4 Codes and Standards

N/A

1.5 Abbreviations and Definitions

Term	Definition
CD	Configuration Database
COM	Component Object Model
CS	Configuration Server
ES	Engineering Specification
GUI	Graphical User Interface
ICD	Interface Control Document
IDL	Interface Definition Language
OPERATOR	The Recalculation GUI user
ProDAS	Professional Data Acquisition System
RG	Recalculation GUI
TRSCDB	Test Result and Sensor Calibration Database
VBS client	The Visual Basic Script Code of a recalculation procedure, that uses the exposed methods/property of the RG server
WDBS	Windows Database Server

2. DESIGN

2.1 Introduction

2.1.1.1 The server shall be an out-of-process server.

2.1.1.2 The server shall be an executable, i.e. it shall have the file extension *.exe*.

2.1.1.3 There shall only be one instance of the server running simultaneously within ProDAS.

2.1.1.4 The server shall be thread-safe, i.e. several clients shall be able to access it simultaneously (although currently there is no use case foreseen with more than one automation client at the same time).

2.1.1.5 The server shall be self-registering by calling it with the argument "/RegServer". Registration will take place when ProDAS is installed.

2.1.1.6 The Recalculation GUI is an application and at the same time is a COM server. The application should be started from the Management GUI or from the command line before any client accesses it.

2.1.1.7 Clients of the Recalculation GUI COM server are restricted to the recalculation procedures. The recalculation procedures shall be started only from the Recalculation GUI. It shall not be possible to run more than one recalculation procedure simultaneously or to run standard recalculation and a recalculation procedure simultaneously.

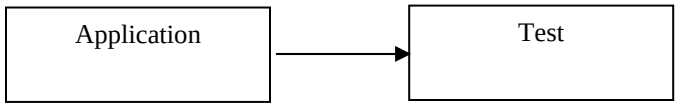
2.1.1.8 Clients of the Recalculation GUI COM server have the possibility to manipulate the graphical user interface of the Recalculation GUI.

2.1.1.9 The Recalculation GUI COM interface should not be used by any client apart from the recalculation procedures.

2.1.1.10 The Recalculation GUI is a COM Client at the same time. Recalculation GUI shall call User Function Engine, External Calculation Program, Print Server, Text Output Page Engine, Configuration Server, User Security System, and Windows Database Server as servers using their COM interfaces.

2.1.1.11 This document defines the COM interface to be used by the clients.

2.1.1.12 The following diagram illustrates the object model.



2.1.1.13 Only the interface “Application” shall be directly accessible. The “Test” interface shall be created by an appropriate Get property of interface Application.

2.1.1.14 There is no configurable time-out for the server. It shall terminate when the Recalculation GUI is terminated by the interactive user.

2.2 Interface "Application"

2.2.1 Design

2.2.1.1 The identification of the interface shall be "proDAS.Recalc".

2.2.1.2 This interface shall be a dispatch interface.

2.2.1.3 This interface shall be an automation interface.

2.2.2 Methods and Properties

2.2.2.1 Method *ShowColumn*

```
//Sets the visibility of a standard column
[id(2), helpstring("Sets the visibility of a standard column")]
HRESULT ShowColumn([in] BSTR columnName);
```

Argument Name	Description
ColumnName	The name associated with the grid’s standard columns: EventID CreationDate TestStep Comment _RecalcSource

2.2.2.1.1 This method displays the given standard column (ES 2.3.5.11).

2.2.2.2 **Method *HideColumn***

```
//Hides a standard column
[id(3), helpstring("Hides a standard column")]
HRESULT HideColumn([in] BSTR columnName);
```

Argument Name	Description
ColumnName	The name associated with the grid's standard columns: EventID CreationDate TestStep Comment _RecalcSource

2.2.2.2.1 This method hides the given standard column (ES 2.3.5.11).

2.2.2.3 **Property *Test***

```
//Provides access to the active test
[propget, id(1), helpstring("Provides access to the active (current)
TEST used by OPERATOR and showed by GUI of RG")]
HRESULT Test([out, retval] ITest *pVal);
```

Argument Name	Description
*pVal	Pointer to the ITest interface

2.2.2.3.1 The get property Test provides access to the active (current) TEST used by the OPERATOR and shown by the GUI of RG. The OPERATOR could already have handled the TEST directly, without using this interface. The TEST already contains the TEST-DATA including the fullsets associated with the current test configuration, the associated channel names and their values, and the corresponding recalculation procedures.

2.2.3 Events Fired

2.2.3.1 There are no events.

2.2.4 Usage Conditions and Restrictions

2.2.4.1 There are no usage restrictions.

2.2.5 Persistent Data

2.2.5.1 The description of the current TEST which has been obtained using the get property *Test* is stored in the Recalculation.ini (RG) file. This test description consists of:

(config ID,engineName, engineStandard, serialNumber, buildNumber, testName)

2.2.6 Example

2.2.6.1 The following is an example of a VBS client.

```
dim app
Set app = CreateObject("proDAS.Recalc")

msgbox "VBS: Put EventID  column non_visible"
app.HideColumn(“EventID”)

msgbox "VBS: Put EventID column visible again"
app.ShowColumn(“EventID”)

msgbox "VBS: Get the active test"
dim test
Set test = app.Test

msgbox "VBS: Final"
```

2.3 Interface "Test"

2.3.1 Design

2.3.1.1 This interface shall not be directly accessible. It shall be accessed through the *Application* interface using the *Test* property.

2.3.1.2 This interface shall be a dispatch interface.

2.3.1.3 This interface shall be an automation interface.

2.3.2 Methods and Properties

2.3.2.1 Property *LastCallFailed*

```
[propget, id(43), helpstring("Returns true if the most recent call to
any method or property has failed")]
HRESULT LastCallFailed([out, retval] BOOL *pVal);
```

Argument Name	Description
*pVal	The error status of the most recent call to any method or property (other than this property itself) of this Test COM object.

2.3.2.2 The success state or failure state of any function call can be retrieved using the property LastCallFailed. The description of several methods and properties of this interface say that these generate an error. In these cases, the property LastCallFailed would return TRUE, else it would return FALSE.

2.3.2.3 Method *Select*

```
// Fills the TEST object with TEST_DATA from TRSCDB
[id(2), helpstring("Fills the current TEST with TEST_DATA from TRSCDB
in according to the parameters")]
HRESULT Select([in] BSTR engineType, [in] BSTR EngineStandard , [in]
BSTR SerialNumber, [in] BSTR BuildNumber, [in] BSTR TestName, [in]
long ConfigID);
```

Argument Name	Description
`EngineType`	Engine type
`EngineStandard`	Engine standard
`SerialNumber`	Serial number
`BuildNumber`	Build number
`TestName`	Test name
`ConfigID`	Test Configuration ID

2.3.2.3.1 The method ***Select*** selects a new test from TRSCDB (ES 3.1.4.7) identified by test description (engineType /engineStandard /serialNumber /buildNumber /testName) and test configuration configID. The description and configuration will be stored into the Recalculation.ini file and into the corresponding drop list combo box. If the selection fails (bad parameters) the existing test object will remain and a warning message will be generated in the Output Message Window of the RG.

Note: When selecting a new test from the TRSCDB the current configuration (default or alternate test configuration) of the Recalculation GUI is not changed. The current test configuration ID is always shown in the window's caption bar.

2.3.2.4 Method *SetChannelFilter*

```
[id(5),  helpstring("Allows you to filter the non-displayed channels
using regular expression.")]
HRESULT SetChannelFilter([in] BSTR patternExpression);
```

Argument Name	Description
`patternExpression`	A regular expression to filter the non-displayed channels. "<ALL>"=Meaning all channel names. The regular expression metacharacters are: **"."** = Matches any single character except "\n". To match any character including the '\n', use a pattern such as '[.\n]'. , so "a.b" matches "aab" and "a3b", but not "ab". "|" = Is used for alternative matching, as in "a|b", which will match either "a" or "b". **"*"** = Matches the preceding character zero or more times, so the regular expression "fo*" matches both "f" and "foo". **"+"** = Matches the preceding character one or more times, so "fo+" matches "foo" but not "f". **"?"** = Matches the preceding subexpression zero or just one time. For example, 'do(es)?' matches the "do" in "do" or "does". '?' is equivalent to '{0,1}'. **"^"** = Indicates a match at the beginning of a string If stands for the beginning of a string, "^i" will match "is" but not "mi". **"$"** = Indicates a match at the end of a string, so "$i" will match "mi" but not "is". **[]** = Brackets are used to express character and digit sets and ranges. For example, the expression "[abcd]" will match any of the enclosed characters in the target string. The whole lower-case alphabet can be expressed using "[a-z]". To include both upper- and lower-case letters in the regular expression, write the expression like this: "[a-zA-Z]". To search for digits in a string, use "[0-9]".

2.3.2.4.1 The method ***SetChannelFilter*** allows you to use a regular expression filter for the non-displayed channels. Setting the regular expression will result in selecting the appropriate channel names in the list of non-displayed channels (ES 3.1.5.27).

2.3.2.5 Method *SetFullsetFilter*

```
[id(3), helpstring("Allows you to set a filter condition to filter the
fullsets of the test object")]
HRESULT SetFullsetFilter([in] BSTR channelName, [in] BSTR Operator,
[in] double Value);
```

Argument Name	**Description**	
`ChannelName`	The name of the channel to be filtered	
`Operator`	Comparation operator The VBS client shall use the following operators:	
	Meaning	*Operator String*
	Equal	=
	Not equal	!= or <> (the operators are equivalent)
	Less than	<
	Less than or equal	<=
	Greater than	>
	Greater than or equal	>=
`Value`	Value (double)	

2.3.2.5.1 The ***SetFullsetFilter*** method allows you to set the filter condition to filter the fullsets (ES 3.1.5.13, 3.1.5.17). Using this method, the filter condition is always re-initialised.

2.3.2.5.2 Defining a fullset filter shall not immediately trigger displaying or re-displaying fullset channel values; display will be delayed until explicitly requested.

2.3.2.5.3 If the fullset does not contain the selected channel, or the operator is not a valid one, an error shall be generated.

2.3.2.6 Method *AddFullsetFilter*

```
[id(34), helpstring("Allows you to add a filter condition to the
existent condition to filter the fullsets of the test object")]
HRESULT AddFullsetFilter([in] BSTR logicOperator, [in] BSTR
channelName, [in] BSTR Operator, [in] double Value);
```

Argument Name	**Description**	
`LogicOperator`	Logic operator (AND, OR)	
`ChannelName`	The name of the channel to be filtered	
`Operator`	Comparation operator The VBS client shall use the following operators:	
	Meaning	*Operator String*
	Equal	=
	Not equal	!= or <> (the operators are equivalent)
	Less than	<
	Less than or equal	<=

Argument Name	Description	
	Greater than	>
	Greater than or equal	>=
Value	Value (double)	

2.3.2.6.1 The ***AddFullsetFilter*** method allows you to add a filter condition to an existing condition to filter the fullsets (ES 3.1.5.13). This method has to be called after ***SetFullsetFilter*** method. If not, an error shall be generated.

2.3.2.6.2 Defining a fullset filter shall not immediately trigger displaying or re-displaying fullset channel values; display will be delayed until explicitly requested.

2.3.2.6.3 If the fullset does not contain the selected channel, or the method was called without setting the initial condition, or the operators are not valid operators an error shall be generated.

2.3.2.7 **Method *ResetFullsetFilter***

```
[id(4), helpstring("Allows you to reset the existing filter conditions
of fullsets")]
HRESULT ResetFullsetFilter();
```

2.3.2.7.1 The method ***ResetFullsetFilter*** allows you to reset the existing filter conditions of fullsets (ES 3.1.5.17). Resetting a fullset filter shall not immediately trigger displaying or re-displaying fullset channel values; display will be delayed until explicitly requested.

2.3.2.8 **Method *DisplayFullset***

```
[id(7),  helpstring("Allows you to display the list of fullsets in the
list according with selected channels and fullsets filter condition")]
HRESULT DisplayFullset();
```

2.3.2.8.1 The method ***DisplayFullset*** (ES 3.1.5.30) allows you to display in the GUI's grid the proper fullsets according to the selected channel and fullset filter conditions. This display (refresh) functionality can also be obtained by pushing the GUI's "Refresh" button.

2.3.2.9 Property *FullsetCount*

```
[propget, id(14), helpstring("Allows you to get the number of fullsets
from the fullset list")]
HRESULT FullsetCount([out, retval] long *pVal);
```

Argument Name	Description
*pVal	Retrieves the number of fullsets from the fullset list. If there are no fullsets, 0 will be return.

2.3.2.9.1 The ***FullsetCount*** get property is taking into consideration if there is any filter condition applied to the fullsets with ***FullsetFilter*** property and the fullset list was refreshed with ***DisplayFullset*** method. The retrieved number is the number of fullsets from the GUI's fullset view panel (Attention: defining a fullset filter will not trigger immediately re-displaying the fullset list; display will be delayed until explicitly requested).

2.3.2.10 Property *EventID*

```
[propget, id(22), helpstring("Allows you to get the event ID of the
current fullset")]
HRESULT EventID([out, retval] int *pVal);
```

Argument Name	Description
*pVal	The Event ID of the current fullset.

2.3.2.11 Method *GotoFirstFullset*

```
[id(15), helpstring("Allows you to set the first fullset from the
displayed fullset list as current fullset")]
HRESULT GotoFirstFullset();
```

If it fails, an error shall be generated.

2.3.2.12 Method *GotoLastFullset*

```
[id(16), helpstring("Allows you to set the last fullset from the
displayed  fullset list as  current fullset ")]
HRESULT GotoLastFullset();
```

If it fails, an error shall be generated.

2.3.2.13 Method *GotoNextFullset*

```
[id(17), helpstring("Allows you to set the next fullset from the
displayed fullset list as  current fullset ")]
HRESULT GotoNextFullset();
```

If it fails, an error shall be generated.

2.3.2.14 **Method *GotoPreviousFullset***

```
[id(18), helpstring("Allows you to set the previous fullset from the
displayed fullset list as  current fullset ")]
HRESULT GotoPreviousFullset();
```

If it fails, an error shall be generated.

2.3.2.15 ***OutputFullsetValues***

```
[id(31), helpstring("Show all channel values in Output Message Window
of the resulting fullset using the Event ID of the source fullset")]
HRESULT OutputFullsetValues([in] long EventID);
```

Argument Name	Description
EventID	The Event ID of the source fullset that was recalculated

2.3.2.15.1 Following the recalculation, the client will be able to retrieve the values of all channels of the resulting fullset using the EventID of the source fullset (ES 3.5.1.15). The channel (initial and recalculated) values will be displayed in the RG-GUI's fullset grid Message Window.

Note: It shall not be necessary to store the recalculated fullset into TRSCDB to retrieve the channel values. This may be used to decide within the recalculation procedure, whether a fullset shall be stored in TRSCDB or not.

2.3.2.16 **Method *Write***

```
[id(24), helpstring("Allows you to send a message to the screen
(Output Message Window of RG-GUI)")]
HRESULT Write([in] BSTR message);
```

Argument Name	Description
Message	The message to be sent to the output message window. (Message Window of RG-GUI, ES 3.5.1.11). If it fails, an error shall be generated.

2.3.2.17 **Method *Print***

```
[id(25), helpstring("Allows you to send a message to Print server ")]
HRESULT Print([in] BSTR message);
```

Argument Name	Description
Message	The message to be sent to the printer (ES 3.5.1.11). If it fails, an error shall be generated.

2.3.2.18 Method *StoreFullset*

```
[id(12), helpstring("Allows you to store a successfully recalculated
fullset")]
HRESULT StoreFullset();
```

2.3.2.18.1 Using the ***StoreFullset*** method, it will be possible to trigger storage of each single successfully recalculated fullset, i.e. let the VBS client decide after recalculation whether to store the current fullset (ES 3.5.1.16).

2.3.2.18.2 To decide after recalculation whether to store a successfully recalculated fullset, it is necessary to switch off writing to the TRSCDB with ***StoreInTRS*** put property.

2.3.2.18.3 If the fullset is not a recalculated fullset or the recalculation of this fullset has failed, nothing happens.

2.3.2.18.4 If the fullset is a recalculated fullset, using the ***TextOutputPage*** and ***Write*** property after ***StoreFullset*** method the user should see in the Message Window the recalculation output values of the current stored fullset. If the fullset was successfully stored, the new Fullset id will be displayed. Otherwise the “No stored in TRDB” message will be displayed.

2.3.2.19 Property *SortChannel*

```
[propget, id(35), helpstring("Get the information that the displayed
channels from second list are sorted or not in alphabetical order")]
HRESULT SortChannel([out, retval] BOOL *pVal);

[propput, id(35), helpstring("Allows you to sort/unsort the displayed
channels in alphabetical order")]
HRESULT SortChannel([in] BOOL newVal);
```

Argument Name	**Description**
`*pVal, newVal`	TRUE –if displayed channels will be/are sorted in alphabetical order, otherwise FALSE The default value is TRUE.

2.3.2.19.1 Using this property the current sort state of the list box with channels foreseen for display can be set and retrieved (ES 3.1.5.23).

2.3.2.20 Property *ShowAllChannels*

```
[propput, id(27), helpstring("Allows you to set all channels from the
first list to be displayed or not")]
HRESULT ShowAllChannels([in] BOOL show);
```

Argument Name	Description
show	If 'show' value is TRUE, all channels from the non-displayed list will be selected to display in the right list box. If 'show' value is FALSE, all channels will appear in the first list box and there are no channels selected to display. Channels not to be displayed will always be sorted alphabetically.

2.3.2.20.1 To select several channels for display, the VBS client will first use the ***SetChannelFilter*** method with the appropriate regular expression and after that will use the ***ShowAllChannels*** property.

2.3.2.20.2 To select all channels for display, the VBS client will first use the ***SetChannelFilter*** method with the "<ALL>" expression and after that will use the ***ShowAllChannels*** property.

2.3.2.20.3 Setting the channels as displayed or not-displayed will not trigger displaying their values. Display will be delayed until explicitly requested.

2.3.2.21 Method *ShowChannel*

```
[id(6), helpstring("Allows you to set a channel to be displayed or
not")]
HRESULT ShowChannel([in] BSTR channelName, [in] BOOL show);
```

Argument Name	Description
channelName	The channel name
Show	If 'show' value is TRUE, the channel will be selected to display. Setting the channels as displayed or not-displayed will not trigger displaying their values, display will be delayed until explicitly requested.

2.3.2.22 Property *SortUp*

```
[propget, id(28), helpstring("Allows you to set a column of fullset-
list to be ascending sorted")]
HRESULT SortUp([out, retval] BSTR *pVal);
```

```
[propput, id(28), helpstring("Get a boolean value if a column of
fullset-list is ascending sorted")]
HRESULT SortUp([in] BSTR newVal);
```

Argument Name	Description
*pVal, newVal	The value shall be the name of a "standard" column or the name of a "channel" column. The 'standard' column names are: EventID, CreationDate, TestStep, Comment, _RecalcSource. A 'channel' column is the column from the fullset_list that contains the name of the channel in the grid's header.

If the value is invalid, nothing happens.

2.3.2.23 Property *SortDown*

```
[propget, id(29), helpstring("Allows you to set/get a column of
Fullset-list to be descending sorted")]
HRESULT SortDown([out, retval] BSTR *pVal);

[propput, id(29), helpstring("Get a boolean value if a column of
fullset-list is descending sorted")]
HRESULT SortDown([in] BSTR newVal);
```

Argument Name	Description
*pVal, newVal	The value shall be the name of a "standard" column or the name of a "channel" column. The 'standard' column names are: EventID, CreationDate, TestStep, Comment, _RecalcSource. A 'channel' column is the column from the fullset_list that contains the name of the channel into the grid's header.

If the value is not valid, nothing happens.

2.3.2.24 Property *Marked*

```
[propput, id(9), helpstring("Allows you to set a fullset as  selected
(marked) fullset for recalculation using the EventID parameter")]
HRESULT Marked([in] long EventID, [in] BOOL select);
```

Argument Name	Description
EventID	A number that defines the fullset's EventID. If EventID parameter is invalid, nothing happens.
Select	If 'select' value is TRUE, the corresponding fullset defined by EventID parameter shall be set as marked, and this fullset will be taken into consideration by "single recalculation-run" (ES 3.1.10.3). If 'select' value is FALSE, the corresponding fullset defined by EventID parameter will be unmarked.

```
[propget, id(9), helpstring("Allows you to get the information if a
fullset identified by EventID parameter is selected (marked) or not")]
HRESULT Marked([in] long EventID, [out, retval] BOOL *pVal);
```

Argument Name	Description
EventID	A number that defines the fullset's EventID. If EventID parameter is invalid, nothing happen.
*pVal	If return value is TRUE, the corresponding fullset defined by EventID parameter is selected and this fullset will be taken into consideration by "single recalculation-run" (ES 3.1.10.3). If return value is FALSE, the corresponding fullset defined by EventID parameter is unmarked.

2.3.2.25 Property *MarkAll*

```
[propput, id(26), helpstring("Allows you to mark/unmark all fullsets
from fullset-list as selected fullsets for recalculation")]
HRESULT MarkAll([in] BOOL select);
```

Argument Name	Description
Select	If 'select' value is TRUE, all displayed fullsets shall be set as marked fullsets and these fullsets will be taken into consideration by "single recalculation-run" (ES 3.1.10.3). If 'select' value is FALSE, all fullsets will be unmarked.

2.3.2.26 Property *ConfigDataFromFullset*

```
[propget, id(10), helpstring("Allows you to get the type of
configuration- fullset-related or corresponding to the current test
configuration")]
HRESULT ConfigDataFromFullset([out, retval]BOOL
*pConfigDataFromFullset);
```

```
[propput, id(10), helpstring("Allows you to set the configuration as
fullset-related or corresponding to the current test configuration ")]
HRESULT ConfigDataFromFullset([in] BOOL configDataFromFullset);
```

Argument Name	Description
`ConfigDataFromFullset/ *pConfigDataFromFullset`	If 'configDataFromFullset' value is TRUE, the configuration data will be retrieved from the TRSCDB. If the required configuration is not available, a warning message will be generated and the ID of the default test configuration will be retrieved instead. If ' configDataFromFullset ' value is FALSE, the current test configuration (which should be the default or an alternate test configuration) will be retrieved.

2.3.2.26.1 ***ConfigDataFromFullset*** property set/get the configuration as fullset-related or corresponding to the current test configuration.

2.3.2.26.2 The corresponding displayed choice ('Config Data from fullset' check button) in RG GUI will not be changed using this property (ES 3.1.9.3).

2.3.2.26.3 The current test configuration ID is always shown in the windows caption bar.

2.3.2.26.4 If the *ConfigDataFromFullset* property is set, this must be done before the first call of the method *RecalculateFullset* in a recalculation procedure to ensure that all fullsets recalculated in a loop will use the same configuration.

2.3.2.27 **Property *StoreInTRS***

```
[propget, id(11), helpstring("Allows you to set/get automatic writing
of recalculated Fullsets into the TRSCDB")]
HRESULT StoreInTRS([out, retval] BOOL *pStoreInTRS);

[propput, id(11), helpstring("Allows you to set/get automatic writing
of recalculated Fullsets into the TRSCDB")]
HRESULT StoreInTRS([in] BOOL storeInTRS);
```

Argument Name	Description
`StoreInTRS/ *pStoreInTRS`	If 'pStoreInTRS' value is TRUE/FALSE, automatic writing to the TRSCDB will be switched ON/OFF. Notes: Writing to the TRSCDB is only possible if the user has write privilege to the TRSCDB.

2.3.2.27.1 The corresponding displayed choice (‘Store in TRDB’ check button) in RG will not be changed using this property.

2.3.2.28 **Property *CopyConfig***

```
[propget, id(33), helpstring("Create a copy of the selected
configuration and retrieve the configuration ID of this copy")]
HRESULT CopyConfig([out, retval] long *pVal);
```

Argument Name	**Description**
*pVal	The retrieved configID. Notes: Without copying the current test configuration the VBS client will not be able to retrieve the configID and will not be able to modify it (ES 3.1.9.10).

2.3.2.28.1 If the *CopyConfig* property is called, this must be done before the first call of the method *RecalculateFullset* in a recalculation procedure to ensure that all fullsets recalculated in a loop will use the same configuration.

2.3.2.29 **Method *RecalculateFullset***

```
[id(19), helpstring("Allows you to execute a 'single recalculation-
run' for the current fullset")] HRESULT RecalculateFullset ();
```

2.3.2.29.1 Using this property, a single fullset may be recalculated (ES 3.1.10.3).

2.3.2.29.2 If the current fullset has not been selected (marked) for recalculation, nothing happens.

2.3.2.30 **Property *ModifyConstantChannel***

```
[propput, id(32), helpstring("Allows you to modify the value of a
constant channel for the current fullset")] HRESULT
ModifyConstantChannel([in] BSTR channelName, [in] double newVal);
```

Argument Name	**Description**
`channelName`	The name of the constant channel that has to be modified.
`newVal`	The new constant channel value that will substitute the old value.

2.3.2.30.1 This property allows the VBS client to modify the value of a constant channel for the selected fullset during each single recalculation run (ES 3.5.1.8). The value provided by the recalculation procedure will replace the value retrieved from TRSCDB. Following the execution of the recalculation run, these modifications will be removed.

2.3.2.30.2 If the channel name is not a valid one or the current fullset is not a marked fullset, an error shall be generated and nothing will happen.

2.3.2.31 **Property *FullsetChannelValue***

```
[propget, id(20), helpstring("Allows you to get the value of a channel
for the current fullset")]
HRESULT FullsetChannelValue([in] BSTR channelName, [out, retval] float
*pVal);
```

Argument Name	**Description**
ChannelName	The channel name
*pVal	The value of the channel. If the property is called after *RecalculateFullset*, the value will be the recalculated value of the channel.

2.3.2.32 **Property *TextOutputPage***

```
[propget, id(23), helpstring("Retrieves a formated page from the Text
Output Page Server based on the recalculation values of the current
selected fullset")]
HRESULT TextOutputPage([in] BSTR pageName, [out, retval] BSTR *pVal);
```

Argument Name	**Description**
pageName	The page name
*pVal	The string that contains the recalculation output values of the current selected fullset based on the given page. If the given page does not exist, an empty string will be returned.

2.3.3 Events Fired

2.3.3.1 There will be no events.

2.3.4 Usage Conditions and Restrictions

2.3.4.1 Selecting a new test from the TRSCDB using the Select method will not change the current configuration (default or alternate test configuration) of the Recalculation GUI.

2.3.4.2 The displayed channels (the second list of the channel view panel) are sorted by default in alphabetical order. The channels should be displayed in the order in which they are selected when *SortChannel* put property is FALSE.

2.3.4.3 The *SetFullsetFilter* method has to be called before calling the *AddFullsetFilter* method. If *AddFullsetFilter* method is called after *ResetFullsetFilter*, the return value will be FALSE. Using the *SetFullsetFilter* method will always re-initialised the filter condition.

2.3.4.4 The VBS client shall set the first fullset before using the iteration loop for displayed fullsets. The first (current) fullset should be set with *GotoFirstFullset* method.

2.3.4.5 The *GotoFirstFullset, GotoNextFullset, GotoPreviousFullset, GotoLastFullset,* methods iterate through the set of all displayed fullsets and set the current fullset. The iteration is taking into consideration the filter conditions applied to the fullsets if the list of fullsets was refreshed with *DisplayFullset* method. If the VBS client sets a fullset filter condition he shall call the *DisplayFullset* method before iterating through the list of fullsets. If the fullset list is sorted according to any column, the iteration sequence will comply to this sorting order. The VBS client should not call the *SortUp* or *SortDown* put properties inside the iteration.

2.3.4.6 The automation client should use the get property *Marked* to check if the current fullset (set with iterating methods) has been marked for recalculation by the interactive user or not. If the methods *RecalculateFullset, ModifyConstantChannel* or *TextOutputPage* are called to an unmarked fullset for recalculation, an error shall be generated and nothing happens.

2.3.4.7 Using the *SortUp* or *SortDown* put properties will deselect the fullsets marked for recalculation.

2.3.4.8 Using the *StoreFullset* method to decide after recalculation whether to store a successfully recalculated fullset, it is necessary to switch off writing to TRSCDB with the *StoreInTRS* property.

2.3.4.9 If the *ConfigDataFromFullset* property is set, this must be done before the first call of the method RecalculateFullset in a recalculation procedure to ensure that all fullsets recalculated in a loop will use the same configuration.

2.3.4.10 If the *CopyConfig* property is called, this must be done before the first call of the method RecalculateFullset in a recalculation procedure to ensure that all fullsets recalculated in a loop will use the same configuration.

2.3.5 Persistent Data

2.3.5.1 There will be no persistent data.

2.3.6 Examples

2.3.6.1 The following is an example of a VBS client which demonstrates the handling of the Select method (ES 3.1.4.7) and also specifies whether the fullset-related configuration data shall take precedence over the current test configuration with the *ConfigDataFromFullset* property (ES 3.1.9.1).

2.3.6.1.1 It is possible to select a test which does not exist. In this case, the existing (current) test object will remain and a warning message will be generated in the Message Window of the RG.

2.3.6.1.2 When selecting a new test from the TRSCDB, the current configuration (default or alternate test configuration) of the Recalculation GUI is not changed. The current test configuration ID is always shown in the windows caption bar.

```
'ICDexample1.vbs
NL=Chr(13) & Chr(10)

dim app
Set app = CreateObject("proDAS.Recalc")

msgbox "VBS: Get the active test"
dim test
Set test = app.Test

msgbox "VBS: Select a test from TRDB with non-existing parameters."
test.Select "RB199", "MK103","SN9000","BN9000","",1235

msgbox "VBS: Select a new test from
TRDSCB(RB199/MK103/SN9000/BN9000/Test_RB/1234)."&NL&"The Configuration
Database contains the configID."
test.Select "RB199", "MK103","SN9000","BN9000","Test_RB",1234

msgbox "VBS:Put ConfigDataFromFullset as true"
test.ConfigDataFromFullset=true
msgbox "The Current ConfigID is fullset related."&NL&"See caption
bar."& NL & "Note: See check box:The corresponding displayed choice
has not been changed."

msgbox "VBS:Put ConfigDataFromFullset as false"
test.ConfigDataFromFullset=false
msgbox "The Current ConfigID is the current test
configuration."&NL&"See caption bar."& NL & "Note: See check box:The
corresponding displayed choice has not been changed."&NL&"See the new
procedure list"

msgbox "VBS: Select a new test from TRDSCB
(RB199/MK103/SN9000/BN9000/Test_RB/1200)."&NL&"The configID is missing
from Configuration Database"
test.Select "RB199", "MK103","SN9000","BN9000","Test_RB",1200
```

```
msgbox "ConfigDataFromFullset is false"
msgbox "The ConfigID is the current test configuration."&NL&"See
caption bar."

msgbox "VBS:Put ConfigDataFromFullset as true"
test.ConfigDataFromFullset=true
msgbox "The Current ConfigID is the current test configuration because
the fullset-related configuration is missing from CS."&NL&"See caption
bar."

msgbox "VBS:Final"
```

2.3.6.2 The following is an example of a VBS client which illustrates how to select a channel for display (ES 3.1.5.28), how the displayed channels are sorted (ES 3.1.5.23), how fullsets are filtered and sorted (ES 3.1.5.17, ES 3.1.5.32), how to iterate through all displayed fullsets and how to mark fullsets for recalculation (ES 3.1.6.2).

2.3.6.2.1 Selecting channels to be displayed or defining (resetting) a fullset filter will not immediately trigger displaying or re-displaying the resulting fullsets in the GUI's grid. This displaying (refreshing) functionality is delayed until explicitly requested by ***DisplayFullset*** method.

```
'ICDexample2.vbs
NL=Chr(13) & Chr(10)

dim app
Set app = CreateObject("proDAS.Recalc")

msgbox "VBS: Get the active test"
dim test
Set test = app.Test
test.DisplayFullset
msgbox "VBS: Select a new test with around 1100 channels."&"Please, be
patient!"
test.Select "PW4000_94","SN100","SN4000","45654655","PerfTest4060_2",1004

msgbox "VBS: Filter (select) all channels"
test.SetChannelFilter "<ALL>"

msgbox "VBS: Put all channels in the second-list"
test.ShowAllChannels=TRUE

msgbox "VBS: Put all channels in the first-list"
test.ShowAllChannels=FALSE

msgbox "VBS: Filter the input channels using WFC.* pattern expression"
test.SetChannelFilter("WFC.*")

msgbox "VBS: Put all filtered channels as displayed channels in the
second-list"
test.ShowAllChannels=TRUE
```

```
msgbox "VBS: Put the displayed channels in the order in which they are
selected"
test.SortChannel=FALSE

msgbox "VBS: Filter the input channels using ADA.* pattern expression"
test.SetChannelFilter("ADA.*")

msgbox "VBS: Put all filtered channels as displayed channels in the
second-list "
test.ShowAllChannels=TRUE

msgbox "VBS: Select the channel WF1 not to be displayed "
test.ShowChannel "WF1", FALSE

msgbox "VBS: Fullset display"
test.DisplayFullset()

msgbox "VBS: Sort descending WFC"
test.SortDown="WFC"

msgbox "VBS:Set Filter WFC>700"
test.SetFullsetFilter("WFC",">",700)
test.DisplayFullset()

msgbox "VBS: Add Filter condition WFC<=3700"
test.AddFullsetFilter("AND","WFC","<=",3700)
test.DisplayFullset()

msgbox "VBS: Reset filter"
test.ResetFullsetFilter
msgbox "VBS: Fullset display to show the all Fullsets"
test.DisplayFullset()

msgbox "VBS: Iterate through the displayed fullsets"
fullsetCount=test.FullsetCount
msgbox "Displayed fullsets number:"&fullsetCount

test.GotoFirstFullset
For i = 1 To fullsetCount
msgBox "VBS: the iterated fullset no=" & i & " is the fullset with
eventID=" & test.EventID
      test.GotoNextFullset
Next

test.SortDown="EventID"

test.GotoLastFullset
For i = 1 To 10
  msgBox "VBS: the iterated fullset has the eventID=" & test.EventID
  test.GotoPreviousFullset
Next

msgbox "VBS: Select Fullsets 1,2"
```

```
test.Marked(1)=TRUE
test.Marked(2)=TRUE

msgbox "VBS: Sort ascending by EventID."
test.SortUp="EventID"

msgbox "VBS: Select all Fullsets"
test.MarkAll=TRUE

msgbox "VBS: Deselect all Fullsets"
test.MarkAll=FALSE

msgbox "VBS:Final"
```

2.3.6.3 There are four examples (ICDexample3.vbs to ICDexample6.vbs) of a VBS client to present the recalculation process of the selected fullsets.

2.3.6.3.1 If only one fullset or if several fullsets are selected and recalculation is started in one loop, RG will execute a single *recalculation run* (defined in ES 3.1.10.3) for each marked fullset.

2.3.6.3.2 Before starting the recalculation process, the VBS client shall create a copy of the current configuration using the ***CopyConfig*** property (ES 3.5.1.1). The new configuration will replace the current configuration ID.

2.3.6.3.3 Following each single recalculation run the VBS client should display the recalculation results using pre-defined text output pages for channel values in the Message Window and should print them (ES 3.5.1.12).

```
'ICDexample3.vbs
NL=Chr(13) & Chr(10)

dim app
Set app = CreateObject("proDAS.Recalc")

msgbox "VBS: Get the active test"
dim test
Set test = app.Test


msgbox "VBS: Filter the input channels using <ALL> patern expression"
test.SetChannelFilter("<ALL>")
test.ShowAllChannels=TRUE

msgbox "VBS: Fullset display"
test.DisplayFullset()

msgbox "VBS: Select fullsets with EventID= '1','2' for recalculation"
test.Marked(1)=TRUE
test.Marked(2)=TRUE
```

```
msgBox "VBS: Iterate through all displayed fullsets"
fullsetCount=test.FullsetCount
test.GotoFirstFullset()

‘ demonstration of erroneous call
test.GotoPreviousFullset
if ( test.LastCallFailed ) then
   msgbox "last call failed"
   ' correct:
   test.GotoFirstFullset
else
   msgbox "last call succeeded"
end if

For i = 1 To FullsetCount
  IsMarked=test.Marked(test.EventID)
  If IsMarked=TRUE then
     Mtext=" marked "
  Else
     Mtext=" unmarked "
  End if
  msgBox "VBS: the iterated fullset no=" & i & " is the fullset with
  eventID=" & test.EventID & " and the fullset is" & Mtext & "for
  recalculation"
  test.GotoNextFullset()
Next

msgBox "VBS: Create a copy of the selected configuration"
configID=test.CopyConfig
msgBox "New configID:"&configID

msgBox "VBS: Execute recalculation and display the results"

test.GotoFirstFullset()
dim outputResultString
For i = 1 To fullsetCount
  IsMarked=test.Marked(test.EventID)
  If  isMarked =TRUE then
    test.RecalculateFullset
    outputResultString=test.TextOutputPage("Page no. 1")
    test.Write outputResultString
    test.Print outputResultString
  end if
  test.GotoNextFullset()

Next

msgbox "Display the recalculated fullsets"
test.DisplayFullset()

msgbox "VBS:Final"
```

2.3.6.3.4 Using the ***ConfigDataFromFullset*** property it is possible to specify whether the fullset-related configuration data shall take precedence over the configuration of the current test configuration or vice versa (ES 3.1.9.3). The VBS client shall use this property before calling the ***CopyConfig*** property.

```
'ICDexample4.vbs

NL=Chr(13) & Chr(10)

dim app
Set app = CreateObject("proDAS.Recalc")

msgbox "VBS: Get the active test"
dim test
Set test = app.Test

msgbox "VBS: Show all channels"
test.SetChannelFilter “ALL”
test.ShowAllChannels = TRUE

msgbox "VBS: Fullset display"
test.DisplayFullset

msgbox "VBS: Select fullsets '1','2'"
test.Marked(1)=TRUE
test.Marked(2)=TRUE

fullsetCount=test.FullsetCount

msgbox "IMPORTANT! The ConfigDataFromFullset property has to be called
before CopyConfig property!"
msgbox "Put ConfigDataFromFullset as FALSE"

b_yes=FALSE
test.ConfigDataFromFullset=b_yes
msgbox "Now, ConfigDataFromFullset is " & test.ConfigDataFromFullset &
NL & "Note: The corresponding displayed choice has not been changed"

msgBox "VBS: Create a copy of the selected configuration"
ID=test.CopyConfig
msgBox "New configID:"&ID

msgBox "VBS: Execute recalculation and display the results"

test.GotoFirstFullset
dim outputResultString

For i = 1 To fullsetCount
  IsMarked=Test.Marked(test.EventID)
  If  isMarked =TRUE then
    test.RecalculateFullset
    outputResultString=test.TextOutputPage("Page no. 1")
    test.Write outputResultString
```

```
    b_yes=Not(b_yes)
      msgbox "Put ConfigDataFromFullset as "& b_yes &NL&"This action
has no
      effect after the Copy of the Configuration was made!"
    test.ConfigDataFromFullset=b_yes
    msgBox "See caption bar: the configuration ID is not changed!"
  end if
  test.GotoNextFullset

Next

msgbox "Display the recalculated fullsets"
test.DisplayFullset

msgbox "VBS:Final"
```

2.3.6.3.5 It is not necessary to store the recalculated fullset into TRSCDB to retrieve the channel values. It is possible to decide within the recalculation procedure, whether a fullset shall be stored in TRSCDB or not using the ***StoreInTRS*** property. Following the recalculation, the client will be able to retrieve the values of all channels of the resulting fullset using the EventID of the source fullset (ES 3.5.1.15) and trigger storage of each single successfully recalculated fullset. The channel values will be displayed in the Output Message Window of RG-GUI.

```
'ICDexample5.vbs
NL=Chr(13) & Chr(10)

dim app
Set app = CreateObject("proDAS.Recalc")

msgbox "VBS: Get the active test"
dim test
Set test = app.Test

msgbox "VBS: Filter the input channels using <ALL> patern expression"
test.SetChannelFilter("<ALL>")
test.ShowAllChannels=TRUE

msgbox "VBS: Fullset display"
test.DisplayFullset()

msgbox "VBS: Select fullsets '1','2'"
test.Marked(1)=TRUE
test.Marked(2)=TRUE

msgBox "VBS: Create a copy of the selected configuration"
configID=test.CopyConfig
msgbox "VBS: The new copied configID:"&configID

'-----------------------------------------------------------------------
----
```

```
'msgbox "VBS: Retrieves the values of channels after recalculation of
all selected fullsets"
'-------------------------------------------------------------------
----
msgBox "VBS: Execute recalculation"

test.GotoFirstFullset()
dim outputResultString

test.StoreInTRS = false
msgBox "VBS: Set store in TRS as false" &NL& "Note: The corresponding
displayed choice has not been changed"

For i = 1 To fullsetCount
  IsMarked=Test.Marked(test.EventID)
  If  isMarked =TRUE then
      test.RecalculateFullset
  end if
  test.GotoNextFullset()

Next

msgBox "VBS: Get channel values for eventID=1"&NL&"See Recalculation
Output Window"
test.OutputFullsetValues(1)
AskMsgBox("Page no. 1")

msgBox "VBS: Get channel values for eventID=2"&NL&"See Recalculation
Output Window"
test.OutputFullsetValues(2)
AskMsgBox("Page no. 2")

msgbox "Display the recalculated fullsets"
test.DisplayFullset()

'-------------------------------------------------------------------
'msgbox "VBS: Retrieves the values of channels after each
recalculatated fullset"
'-------------------------------------------------------------------
msgbox "VBS: Sort by eventID"
test.SortUp="EventID"

msgbox "VBS: Store in TRS is already false"

msgbox "VBS: Select fullsets”
test.Marked(1)=TRUE
test.Marked(2)=TRUE

fullsetCount=test.FullsetCount

msgBox "VBS: Execute recalculation and display the results"

test.GotoFirstFullset()
```

```
For i = 1 To fullsetCount
  IsMarked=Test.Marked(test.EventID)
  If  isMarked =TRUE then
      test.RecalculateFullset
      msgBox "VBS: Get channel values for eventID="&i&NL&"See
      Recalculation Output Window"
      test.OutputFullsetValues(i)
      AskMsgBox("Page no. 1")
  End if
  test.GotoNextFullset()

Next

msgbox "Display the recalculated fullsets"
test.DisplayFullset()

msgbox "VBS:Final"

'---------------------------------------------------------------------
' Ask if the user wants to save the recalculated fullset
Sub AskMsgBox (sMessage)
      MyVar = MsgBox ("Store the recalculated fullset", 65,
    "TestStoreFullset")
      if (Myvar=1)  then
            test.StoreFullset
            outputResultString=test.TextOutputPage(sMessage)
            test.Write outputResultString
      end if
end sub
'---------------------------------------------------------------------
```

2.3.6.3.6 During each single recalculation run for the selected fullset the VBS client shall be able to replace the value of constant channels using ***ModifyConstantChannel*** property (ES 3.5.1.8). The value provided by the VBS client will replace the value retrieved from TRSCDB. Following the execution of the recalculation run, these modifications will be removed.

```
'ICDexample6.vbs

NL=Chr(13) & Chr(10)

dim app
Set app = CreateObject("proDAS.Recalc")

msgbox "VBS: Get the active test"
dim test
Set test = app.Test

msgbox "VBS: Select a new test from TRDSCB"
test.Select "RB199", "MK103","SN9000","BN9000","Test_RB",1235

msgbox "VBS: Filter the input channels using <ALL> patern expression"
```

```
test.SetChannelFilter("<ALL>")
test.ShowAllChannels=TRUE

msgbox "VBS: Fullset display"
test.DisplayFullset()

msgbox "VBS: Select fullsets '1','2'"
test.Marked(1)=TRUE
test.Marked(2)=TRUE

fullsetCount=test.FullsetCount

msgBox "VBS: Create a copy of the selected configuration"
configID=test.CopyConfig
msgbox "VBS: The new copied configID:"&configID

msgBox "VBS: Execute recalculation and display the results"

test.GotoFirstFullset()
        IsMarked=Test.Marked(test.EventID)
If  isMarked =TRUE then
    msgbox "Modify constant channels"
    test.ModifyConstantChannel(“KL")="240.90"
end if

dim outputResultString

msgbox "VBS: Put StoreInTRS false"
test.StoreInTRS = false

For i = 1 To fullsetCount

  IsMarked=Test.Marked(test.EventID)
  If  isMarked =TRUE then
      ch1=test.FullsetChannelValue("KL")
      test.RecalculateFullset
      outputResultString=test.TextOutputPage("Page no. 1")
      test.Write outputResultString
      msgBox "VBS: Recalculation for eventID="&test.EventID
      ch2=test.FullsetChannelValue("KL")
      eventID=test.EventID
      msgbox "Event ID:"&EventID&" Channel KL"&NL&"Initial value:"&
ch1
      &NL&"Recalculated value:"& ch2
      if test.StoreInTRS=FALSE then
          MyVar = MsgBox ("Store recalculated fullset", 65,
                          "TestStoreFullset")
          if Myvar=1  then
               test.StoreFullset
               outputResultString=test.TextOutputPage("Page no. 2")
               test.Write outputResultString
          end if
      end if
  end if
```

```
  test.GotoNextFullset()

Next

msgbox "Display the recalculated fullsets"
test.DisplayFullset()

msgbox "VBS:Final"
```