AERO SUPPORT CORPORATION

MTU Aero Engines

# COM INTERFACE CONTROL DOCUMENT FOR

## Configuration Server

DATE: 11/09/12

MDS RELEASED DOCUMENT

AUTH: E

| DOCUMENT NUMBER | REVISION |
|---|---|
| ICD78031.2661 | 7 |

| | NAME | POSITION | COMPANY | SIGNATURE | DATE |
|---|---|---|---|---|---|
| Prepared By: | Martin Hadaller | Software Developer | MTU | [illegible] | 10.09.12 |
| Reviewed By: | Dr. Thomas Speer | Representative Testbed Software | MTU | [illegible] | 10.9.12 |
| Reviewed By: | Gunnar Weber | Senior Manager Testbed Systems | MTU | [illegible] | 10.9.12 |
| Approved By: | France Martel | Product Manager | MDS | [illegible] | 11.9.12 |

**MDS Aero Support Corporation**

1220 Old Innes Road, Suite 200, Ottawa, Ontario, Canada K1B 3V3

Phone (613) 744-7257 Fax (613) 744-8016

**MTU Aero Engines GmbH**

Dachauer Straße 665,

80995 München

**Proprietary Notice**

This document is the property of MDS Aero Support Corporation, and is provided on condition that it be used exclusively for evaluation purposes. Any duplication or reproduction, in whole or in part, without prior written consent of an authorized MDS Aero Support Corporation representative is prohibited.

## Revision History

| Rev. | Date | Author | Sections Affected | Description |
|---|---|---|---|---|
| 6 | Mar 2009 | Kai Scheffler | Rev. Hist. | Added an initial Revision History table |
| 6 | Mar 19, 2009 | Kai Scheffler | 2.1.1.11<br>2.41<br>2.42 | Review of Site Specific Channel Attributes |
| 6 | Aug 28, 2009 | Martin Hadaller | 2.40.2, 2.40.3 | IOptions: SaveAll added, declared IsLocalWrite as obsolete |
| 7 | Sep 17, 2010 | Martin Hadaller | 2.7.3.6 | IErrors: Added GetErrorOrWarning |
| 7 | Dec 2, 2010 | Peter Buchholz | 2.12.3.29 | Added CustomerSensor |
| 7 | Jan 10, 2011 | Peter Buchholz | 2.43,2.44, 2.45, 2.46, 2.472.48, 2.49, 2.50 | Added type descriptions to be used by Tabular Channel Display |
| 7 | Mar 15, 2012 | Martin Hadaller | 2.12.3.22 | Restrict max channel name lenth to 39 characters |
| 7 | Jul 15, 2011 | Martin Hadaller | 2.17.3.8<br>2.17.3.9 | Added new PBS methods |

## TABLE OF CONTENTS

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

# 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 Configuration Server.

1.1.2 The Configuration Server provides access to configuration data, which are stored in the configuration database as XML files.

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

1.1.4 The subsections *Events Fired*, *Usage Conditions and Restrictions*, *Persistent Data* and *Example* of the interface sections will be omitted if there is no relevant content.

1.1.5 For the sake of manageability, we have split the ICD for Configuration Server documentation in three parts. This document describes all interfaces of the CS except for the Subsystem’s specific interfaces of the Channels (c.f. Interface Control Document for the Configuration Server-Channels) and the Subsystems (c.f. Interface Control Document for the Configuration Server-Subsystems).

## 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 |
| ES78031.2660 | Engineering Specification for Configuration Server |
| ES78024.2655 | Engineering Specification for User Functions |
| ES78024.2673 | Interface Control Document for the User Security System |
| ES78031.2801 | Interface Control Document for the Configuration Server-Subsystems |
| ES78031.2782 | Interface Control Document for the Configuration Server-Channels |

## 1.4 Codes and Standards

| Number | Title |
|---|---|
|  | ICD Template |

## 1.5 Abbreviations and Definitions

| Term | Definition |
|---|---|
| May | An option or permission |
| Shall | A mandatory requirement |
| Should | A recommendation |
| Will | A statement of intent |
| MDS | MDS Aero Support Corporation |
| MTU | MTU Aero Engines |
| BSTR | data type (32-bit character pointer) |
| C | C programming language |
| CG | Configuration GUI |
| COM | Component Object Model |
| CS | Configuration Server |
| DCOM | Distributed Component Object Model |
| ES | Engineering Specification |
| EU | Engineering Units |
| FRD | Functional Requirements Document |
| GUI | Graphical User Interface |
| HP | Hewlett-Packard |
| ICD | Interface Control Document |
| IDL | Interface Definition Language |
| LSB | Least Significant Bit |
| MG | Management GUI |
| MSB | Most Significant Bit |
| proDAS | Professional Data Acquisition System |
| ROC | Rate of Change |
| RTE | Real Time Engine |
| TRSCDB | Test Result and Sensor Calibration Database |
| VBS | Visual Basic Script |
| XML | Extensible Mark-up Language |
| XSL | Extensible Stylesheet Language |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

# 2. DESIGN

## 2.1 Introduction

### 2.1.1 Overview

2.1.1.1 The server shall be accessible over the network via DCOM.

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

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

2.1.1.4 There shall only be one instance of the server running simultaneously within proDAS.

2.1.1.5 The server shall be thread-safe, i.e. several clients shall be able to access it simultaneously.

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

2.1.1.7 The server shall have a time-out of 5 seconds, i.e. it shall terminate 5 seconds after the last connection has been destroyed.

2.1.1.8 The keywords used by the Configuration Server shall be case sensitive.

2.1.1.9 There will be no standard interfaces implemented other than *IUnknown* and *IDispatch*, which is implemented with every COM interface.

2.1.1.10 Only the interface *Application* shall be directly accessible. All other interfaces shall be created directly or indirectly from an *Application* interface.

2.1.1.11 The following diagram illustrates the object model, which the clients have to use. An arrow from A to B in the diagram means, that an object with interface B is created by an appropriate get property of interface A.

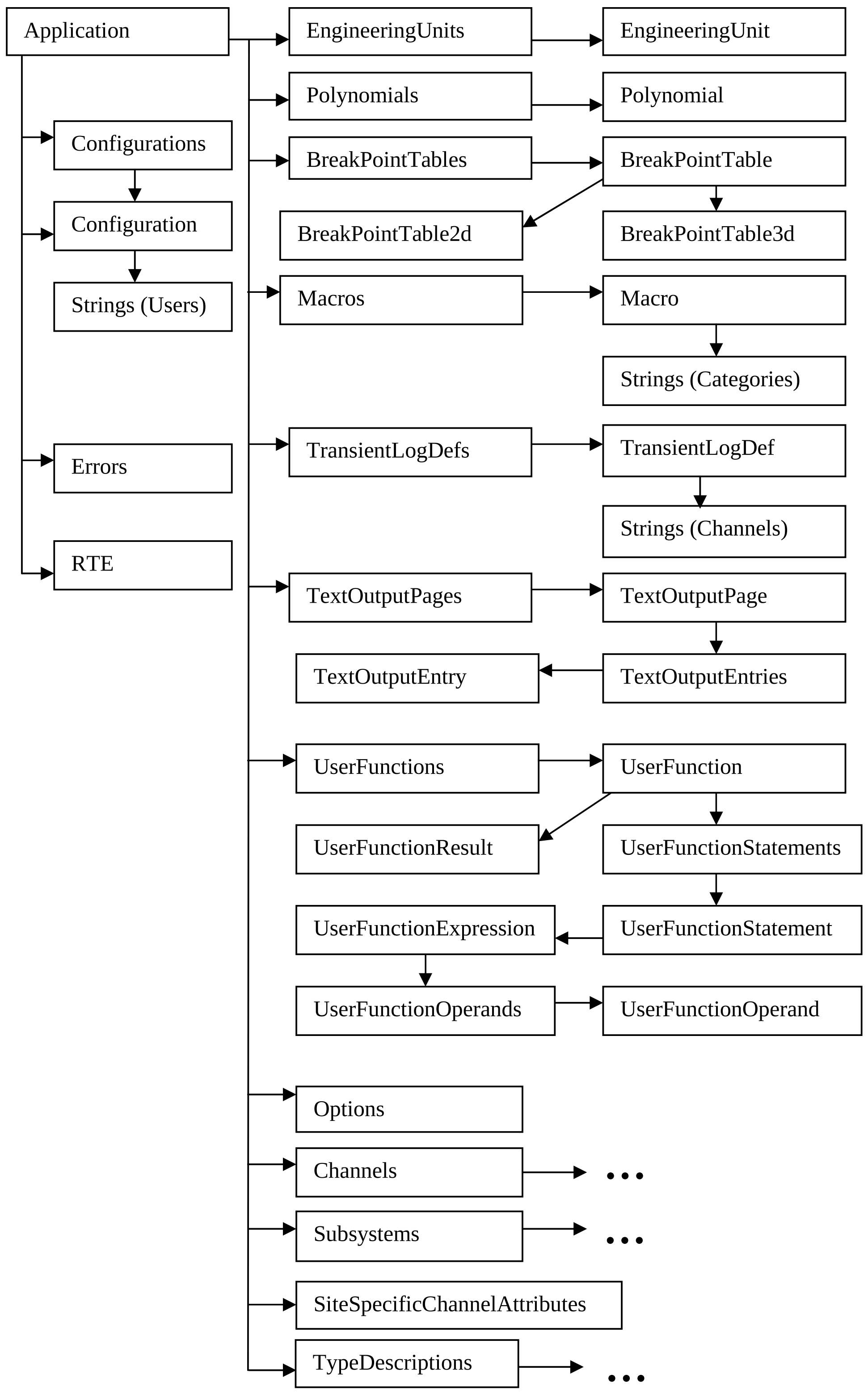

Figure 1: Overall Object Model

2.1.1.12 The interfaces that can be accessed via the Channels and the Subsystem interface are described in the Interface Control Document for the Configuration Server-Channels and the Interface Control Document for the Configuration Server-Subsystems respectively.

### 2.1.2 Generating Errors

2.1.2.1 When it is stated in the description of a property or a method that **an error is generated** this means that a return value unequal to S_OK is returned. In a VBS script this results in an exception to be thrown, unless exception throwing is switched off. A C++ client can read the return value.

### 2.1.3 Configuration Hierarchy

2.1.3.1 According to the Functional Requirements Document for proDAS , there is a hierarchy of configurations. In this hierarchy there are the following **parent child** relations between configurations.

| **Parent** | **Children** |
|---|---|
| Root | *Engines*, *TestCells* |
| Engines | All engine types |
| Test Cells | All test cells |
| Engine Type A | All engine standards belonging to engine type A<br>All pairs (test cell, engine type) with A as engine type |
| Engine Standard B | All customers for engine standard B |
| Customer C | All test configurations for customer C |
| Test cell D | All pairs (test cell, engine type) with D as test cell |
| Pair (engine type A, test cell D) | All test configurations for engine type A in test cell D |

2.1.3.2 For a specific test configuration this results in the following diagram.

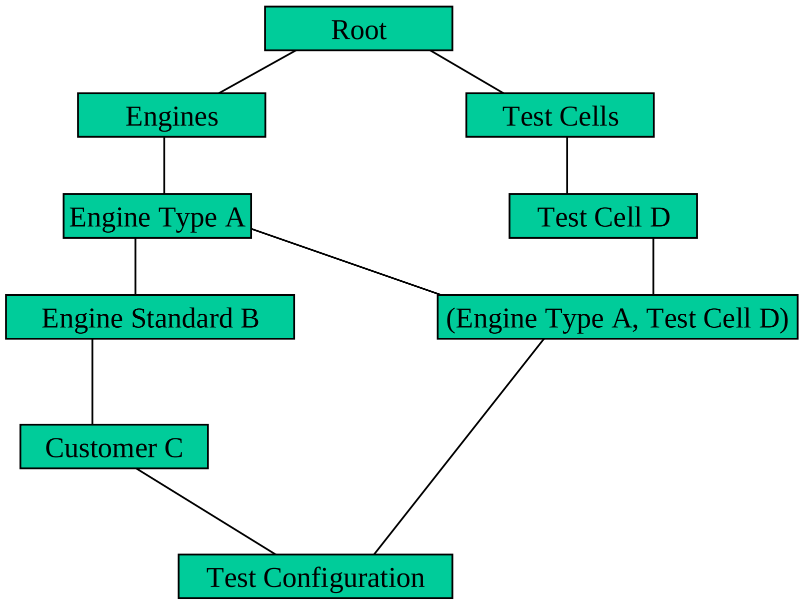

Figure 2: Overall Object Model

2.1.3.3 Note that every configuration will have a parent configuration, except Root, and that every configuration can have child configurations, except the Test Configurations.

2.1.3.4 In many interfaces, representing data records, a property *ConfigLevel* will be defined. This will assign the data record to the current configuration or one of its parent configurations in the following way.

0 = root configuration
1 = test cell configuration
2 = engine type configuration
3 = configuration of an engine type assigned to a test cell
4 = engine standard configuration
5 = customer configuration
6 = test configuration

The configurations *Engines* and *Test Cells* will normally not contain configuration data but there are cases where test cell or engine data are valid for all test cells or all engines.

Note that only the current configuration and its parent configurations will be defined.

## 2.2 Collection Interface

### 2.2.1 Description

2.2.1.1 This interface will be used for all collections. It is the standard method to implement automation collections.

### 2.2.2 Design

2.2.2.1 This interface shall be a dispatch interface.

2.2.2.2 This interface shall be an automation interface.

### 2.2.3 Methods and Properties

2.2.3.1 Property Count

```
// Number of objects
[propget, id(1001)]
HRESULT Count([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | number of objects in the collection. |

2.2.3.2 Method Delete

```
// Delete an item in the collection.
[id(1004)]
HRESULT Delete([in] long Index);
```

| **Argument Name** | **Description** |
|---|---|
| Index | Delete the item with index *Index*. |

### 2.2.3.3 Property _NewEnum

```
// Accessing the enumeration interface
[propget, id(DISPID_NEWENUM)]
HRESULT _NewEnum([out, retval] LPUNKNOWN *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | an interface *IEnumVARIANT*, used for enumerating the members of the collection. |

### 2.2.3.4 Property Item

```
// Accessing an item by index
[propget, id(DISPID_VALUE)]
HRESULT Item([in] long Index, [out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Index | Index, where indexing shall start with 1. If *Index* is not between 1 and *Count* an error *E_INVALIDARG* shall be generated. |
| *pVal | the item with the index *Index*. |

### 2.2.3.5 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

2.2.3.6 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | Indication whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

2.2.3.7 Property New

```
// Create a data record.
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | The index of the newly created data record in the collection. |

2.2.3.8 Method Clear

```
// Delete all items in the collection and remove them.
[id(1008)]
HRESULT Clear();
```

2.2.3.9 Method Clone

```
// Clone the collection object, i.e. make an identical copy.
[id(1009)]
HRESULT Clone([in] LPDISPATCH pClone);
```

| Argument Name | Description |
|---|---|
| pClone | the object to be cloned. |

### 2.2.4 Usage Conditions and Restrictions

2.2.4.1 The type *LPDISPATCH* in *Item* may be replaced by a more specific type.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.3 Interface "Strings"

### 2.3.1 Description

2.3.1.1 This interface will represent a collection of strings.

### 2.3.2 Design

2.3.2.1 This interface shall be a dispatch interface.

2.3.2.2 This interface shall be an automation interface.

### 2.3.3 Methods and Properties

2.3.3.1 Property Count

```
// Number of Strings
[propget, id(1)]
HRESULT Count([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | number of strings in the collection. |

2.3.3.2 Property _NewEnum

```
// Accessing the enumeration interface
[propget, id(DISPID_NEWENUM)]
HRESULT _NewEnum([out, retval] LPUNKNOWN *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | an interface *IEnumVARIANT*, used for enumerating the members of the collection. |

### 2.3.3.3 Property Item

```
// Accessing an item by index
[propget, id(DISPID_VALUE)]
HRESULT Item([in] long Index, [out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Index | Index, where indexing shall start with 1. If *Index* is not between 1 and *Count* an error *E_INVALIDARG* shall be generated. |
| *pVal | the item with the index *Index*. |

### 2.3.3.4 Method Add

```
// Appending an item
[id(4)]
HRESULT Add([in] BSTR Item);
```

| **Argument Name** | **Description** |
|---|---|
| Item | A string. If the item is already present no error is generated. |

### 2.3.3.5 Method Remove

```
// Removing an item
[id(5)]
HRESULT Remove([in] BSTR Item);
```

| **Argument Name** | **Description** |
|---|---|
| Item | A string. If it is not found nothing shall happen. |

### 2.3.3.6 Property IsPresent

```
// Check if an item is contained in the collection
[propget, id(9)]
HRESULT IsPresent([in] BSTR Item, [out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Item | A string the presence of which is to be checked |

| | |
|---|---|
| *pVal | Flag indicating whether the item is in the collection. The string comparison is case insensitive. |

#### 2.3.3.7 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(12)] HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed since the data have been written back to the strings collection the last time. It shall initially be set to *false*. |

#### 2.3.3.8 Method Clone

```
// Clone the collection object, i.e. make an identical copy.
[id(13)]
HRESULT Clone([in] LPDISPATCH pClone);
```

| **Argument Name** | **Description** |
|---|---|
| pClone | the object to be cloned. |

### 2.3.4 Example

2.3.4.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interface *Users*.

```
' Establish the connection to the server.
Set App = CreateObject("proDAS.Config")

' Set the default configuration to the root configuration
set Configs = App.Configurations
idx = Configs.New("","","","",0)
set Config = Configs(idx)
App.DefaultConfiguration = config

' Get the collection of user names.
Set Users = Config.Users

' Display the list of users.
for each user in Users
    message = message & user & vbLf
next
MsgBox message
```

## 2.4 Interface "Application"

### 2.4.1 Description

2.4.1.1 This interface will be the only one of the Configuration Server which can be created directly. Therefore, each client using the Configuration Server will first have to create such an interface.

2.4.1.2 The identification of the interface shall be "proDAS.Config".

2.4.1.3 This interface offers some administration functions and indirectly gives access to all configuration data.

### 2.4.2 Design

2.4.2.1 This interface shall be a dispatch interface.

2.4.2.2 This interface shall be an automation interface.

### 2.4.3 Methods and Properties

2.4.3.1 Property DefaultConfiguration

```
// Default configuration
[propget, id(1)]
HRESULT DefaultConfiguration([out, retval] LPDISPATCH *pVal);
[propput, id(1)]
HRESULT DefaultConfiguration([in] LPDISPATCH newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the default configuration, which is assigned to an *Application* interface as *CurrentConfiguration* property when it is created. |

2.4.3.1.1 This property shall be global, i.e. it is the same for all Application interfaces. If *DefaultConfiguration* is read, a copy shall be made, i.e. modifying the created interface shall not alter the default configuration. This shall only happen if the *DefaultConfiguration* property is written.

### 2.4.3.2 Property DefaultConfigurationId

```
// Default configuration Identifier
[propget, id(2)]
HRESULT DefaultConfigurationId([out, retval] long *pVal);
[propput, id(2)]
HRESULT DefaultConfigurationId([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Id of the default configuration. |

2.4.3.2.1 This property shall be global. Setting it shall also change the *DefaultConfiguration* property, i.e. it is an alternate method of setting the default configuration.

### 2.4.3.3 Property CurrentConfiguration

```
// Configuration to be accessed.
[propget, id(3)]
HRESULT CurrentConfiguration([out, retval] LPDISPATCH *pVal);
[propput, id(3)]
HRESULT CurrentConfiguration([in] LPDISPATCH newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration local to the *Application* interface. |

2.4.3.3.1 If *CurrentConfiguration* is read a copy shall be made, i.e. modifying the created interface shall not alter the configuration. This shall only happen if the *CurrentConfiguration* property is written.

### 2.4.3.4 Property CurrentConfigurationId

```
// Configuration Id to be accessed.
[propget, id(4)]
HRESULT CurrentConfigurationId([out, retval] long *pVal);
[propput, id(4)]
HRESULT CurrentConfigurationId([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Id of the configuration local to the *Application* interface. |

2.4.3.4.1 Setting it shall also change the *CurrentConfiguration* property, i.e. it is an alternate method of setting the current configuration assigned to the *Application* interface.

2.4.3.5 Property Configurations

```
// Collection of configurations
[propget, id(5)]
HRESULT Configurations([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the collection of all configurations. |

2.4.3.6 Property IsAllConsistent

```
// Check of all configuration data
[propget, id(6)]
HRESULT IsAllConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true*, the success of the operation, otherwise *false*. |

2.4.3.6.1 It checks the configuration data of the current configuration and its parent configurations for consistency and shall check the completeness of these configuration data. The consistency check shall be made in the same way as in the *IsConsistent* methods of the single categories of configuration data.

### 2.4.3.7 Property UseInactive

```
// Flag whether inactive data records are to be accessed
[propget, id(8)]
HRESULT UseInactive([out, retval] BOOL *pVal);
[propput, id(8)]
HRESULT UseInactive([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | indicate whether inactive data records are accessed, i.e. if they appear in the collections returned by the appropriate properties of the *Application* interface. This property shall be local. The default value shall be *false.* |

### 2.4.3.8 Property Errors

```
[propget, id(9)]
HRESULT Errors([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | an *Errors* interface. |

### 2.4.3.9 Property EngineeringUnits

```
[propget, id(10)]
HRESULT EngineeringUnits([out, retval] LPDISPATCH *pVal);
[propput, id(10)]
HRESULT EngineeringUnits([in] LPDISPATCH newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Engineering Units configuration data of the current configuration assigned to the *Application* interface. |

2.4.3.9.1 This property shall be local. When this property is read a **copy** shall be made, i.e. changing the created interface shall not alter the respective property. This shall only happen if the property is set. The property *IsAllConsistent* will validate the changed data in the local copy.

### 2.4.3.10 Property Polynomials

```
[propget, id(11)]
HRESULT Polynomials([out, retval] LPDISPATCH *pVal);
[propput, id(11)]
HRESULT Polynomials([in] LPDISPATCH newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Polynomials configuration data of the current configuration assigned to the *Application* interface. |

2.4.3.10.1 This property shall be local. When this property is read a **copy** shall be made, i.e. changing the created interface shall not alter the respective property. This shall only happen if the property is set. The property *IsAllConsistent* will validate the changed data in the local copy.

### 2.4.3.11 Property BreakPointTables

```
[propget, id(12)]
HRESULT BreakPointTables([out, retval] LPDISPATCH *pVal);
[propput, id(12)]
HRESULT BreakPointTables([in] LPDISPATCH newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Break Point Tables configuration data of the current configuration assigned to the *Application* interface. |

2.4.3.11.1 This property shall be local. When this property is read a **copy** shall be made, i.e. changing the created interface shall not alter the respective property. This shall only happen if the property is set. The property *IsAllConsistent* will validate the changed data in the local copy.

### 2.4.3.12 Property Channels

```
[propget, id(13)]
HRESULT Channels([out, retval] LPDISPATCH *pVal);
[propput, id(13)]
HRESULT Channels([in] LPDISPATCH newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the Channels configuration data of the current configuration assigned to the *Application* interface. |

2.4.3.12.1 This property shall be local. When this property is read a **copy** shall be made, i.e. changing the created interface shall not alter the respective property. This shall only happen if the property is set. The property *IsAllConsistent* will validate the changed data in the local copy.

### 2.4.3.13 Property Options

```
// Access to the options
[propget, id(14)]
HRESULT Options([out, retval] LPDISPATCH *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | a reference to the options. |

### 2.4.3.14 Property Macros

```
[propget, id(15)]
HRESULT Macros([out, retval] LPDISPATCH *pVal);
[propput, id(15)]
HRESULT Macros([in] LPDISPATCH newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the Macros configuration data of the current configuration assigned to the *Application* interface. |

2.4.3.14.1 This property shall be local. When this property is read a **copy** shall be made, i.e. changing the created interface shall not alter the respective property. This shall only happen if the property is set. The property *IsAllConsistent* will validate the changed data in the local copy.

### 2.4.3.15 Property TransientLogDefs

```
[propget, id(16)]
HRESULT TransientLogDefs([out, retval] LPDISPATCH *pVal);
[propput, id(16)]
HRESULT TransientLogDefs([in] LPDISPATCH newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Transient Log Definitions configuration data of the current configuration assigned to the *Application* interface. |

2.4.3.15.1 This property shall be local. When this property is read a **copy** shall be made, i.e. changing the created interface shall not alter the respective property. This shall only happen if the property is set. The property *IsAllConsistent* will validate the changed data in the local copy.

### 2.4.3.16 Property TextOutputPages

```
[propget, id(17)]
HRESULT TextOutputPages([out, retval] LPDISPATCH *pVal);
[propput, id(17)]
HRESULT TextOutputPages([in] LPDISPATCH newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Text Output Pages configuration data of the current configuration assigned to the *Application* interface. |

2.4.3.16.1 This property shall be local. When this property is read a **copy** shall be made, i.e. changing the created interface shall not alter the respective property. This shall only happen if the property is set. The property *IsAllConsistent* will validate the changed data in the local copy.

2.4.3.17 Property UserFunctions

```
[propget, id(18)]
HRESULT UserFunctions([out, retval] LPDISPATCH *pVal);
[propput, id(18)]
HRESULT UserFunctions([in] LPDISPATCH newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the User Functions configuration data of the current configuration assigned to the *Application* interface. |

2.4.3.17.1 This property shall be local. When this property is read a **copy** shall be made, i.e. changing the created interface shall not alter the respective property. This shall only happen if the property is set. The property *IsAllConsistent* will validate the changed data in the local copy.

2.4.3.18 Property Subsystems

```
[propget, id(19)]
HRESULT Subsystems([out, retval] LPDISPATCH *pVal);
[propput, id(19)]
HRESULT Subsystems([in] LPDISPATCH newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Subsystems configuration data of the current configuration assigned to the *Application* interface. |

2.4.3.18.1 This property shall be local. When this property is read a **copy** shall be made, i.e. changing the created interface shall not alter the respective property. This shall only happen if the property is set. The property *IsAllConsistent* will validate the changed data in the local copy.

## 2.4.3.19 Method Identify

```
// Identification of the user
[id(20)]
HRESULT Identify([in] BSTR UserName, [in] BSTR Password);
```

| **Argument Name** | **Description** |
|---|---|
| UserName | The user's name. |
| Password | The password associated to the user name. |

2.4.3.19.1 This method shall retrieve the user level from the user security system (cf. Interface Control Document for the User Security System). If this is not successful an error shall be generated and the user level shall be set to zero, i.e. only read access shall be possible.

## 2.4.3.20 Property RTE

```
// Binary data for the RTE
[propget, id(21)]
HRESULT RTE([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | an interface *RTE*, which will supply the data for the RTE in binary form. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.4.3.21 Property Copy

```
// Copy a configuration including all child configurations
// and data
[propget, id(22)]
HRESULT Copy([in] long ConfigLevel, [in] BSTR NewConfig,
             [in] BOOL Recursive, [out, retval] long* pVal);
```

| **Argument Name** | **Description** |
|---|---|
| ConfigLevel | the configuration to be copied as the current configuration (if *ConfigLevel* is the same) or one of its parent configurations (if *ConfigLevel* is less than the configuration level of the current configuration). If *ConfigLevel* has a value less than 1 or greater than the configuration level of the current configuration an error shall be generated. |
| NewConfig | The new configuration shall have the name *NewConfig* and shall have the same parents as the source configuration. |
| Recursive | specifies if the copy operation includes the sub-configuration (child) levels or not. |
| *pVal | the index of the newly created configuration. |

### 2.4.3.22 Property IsLocked

```
// Locking and unlocking data for writing
[propget, id(23)]
HRESULT IsLocked([in] BSTR Category, [in] long ConfigLevel,
      [out, retval] BOOL *pVal);
[propput, id(23)]
HRESULT IsLocked([in] BSTR Category, [in] long ConfigLevel,
      [in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Category | The category to be (un)locked, i.e. BreakPointTables, Channels, EngineeringUnits, Macros, Polynomials, Subsystems, TextOutputPages, TransientLogDefs, and UserFunctions. |
| ConfigLevel | The configuration level to be (un)locked, where –1 means all levels shall be locked. An error shall be returned if the configuration level is outside of the following range: -1 to 6. |
| *pVal, newVal | represents the locking status of the data of the category *Category* and the configuration level *ConfigLevel* for writing. Locking shall be performed by setting the property to *true*, unlocking by setting it to *false*. The default value shall be *false*. |

2.4.3.22.1 Editing of configuration data will normally be performed in the following way:

- Lock the data to be edited.
- Read the data from the Configuration server.
- Edit the data.
- Write the data back to the Configuration Server.
- Unlock the data.

### 2.4.3.23 Method Rename

```
// Rename a configuration level.
[id(24)]
HRESULT Rename([in] long ConfigLevel, [in] BSTR NewName);
```

| **Argument Name** | **Description** |
|---|---|
| ConfigLevel | The level *ConfigLevel* of the configuration to be renamed. |
| NewName | The new name *NewName* of the configuration. |

### 2.4.3.24 Property Category

```
// Returns the category associated to a configuration file.
[propget, id(25)]
HRESULT Category([in] BSTR FileName,
      [out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | The name of the fully qualified file *FileName* from which the category is to be retrieved. |
| *pVal | The category name associated to the file *FileName*, i.e. BreakPointTables, Channels, EngineeringUnits, Macros, Polynomials, Subsystems, TextOutputPages, TransientLogDefs, or UserFunctions. If the file does not exist or if the file is not a configuration file containing a valid category, an empty string is returned. |

2.4.3.24.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.4.3.25 Property CategoryXML

```
// Returns the category contained within an XML string.
[propget, id(30)]
HRESULT CategoryXML([in] VARIANT XML,
      [out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| XML | XML data as a byte array. |
| *pVal | The category name that is contained within the XML data, i.e. BreakPointTables, Channels, EngineeringUnits, Macros, Polynomials, Subsystems, TextOutputPages, TransientLogDefs, or UserFunctions. If the XML data does not contain a valid category, an empty string is returned. |

### 2.4.3.26 Property DataRecordsCount

```
// Returns the number of data records contained in a file
[propget, id(26)]
HRESULT DataRecordsCount([in] BSTR FileName,
      [in] BSTR Category, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | The name of the fully qualified file *FileName* from which the record count is to be retrieved. |
| Category | The category associated to the file *FileName*, i.e. BreakPointTables, Channels, EngineeringUnits, Macros, Polynomials, Subsystems, TextOutputPages, TransientLogDefs, or UserFunctions. |
| *pVal | The number of records of the category *Category* contained in the file *FileName*. If the file does not exist or the category is not valid, an error shall be generated. |

2.4.3.26.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.4.3.27 Property DataRecordsCountXML

```
// Returns the number of data records contained
// in an XML string.
[propget, id(29)]
HRESULT DataRecordsCountXML([in] VARIANT XML,
      [in] BSTR Category, [out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| XML | XML data as a byte array. |
| Category | The category associated to the XML data, i.e. BreakPointTables, Channels, EngineeringUnits, Macros, Polynomials, Subsystems, TextOutputPages, TransientLogDefs, or UserFunctions. |
| *pVal | The number of records of the category *Category* contained in the XML data. If the category is not valid, an error shall be generated. |

### 2.4.3.28 Property NewCollection

```
// Create a new empty collection of data records of a category.
[propget, id(27)]
HRESULT NewCollection([in] BSTR Category,
                      [out, retval] LPDISPATCH *pVal);
```

| Argument Name | Description |
|---|---|
| Category | Name of the data category. |
| *pVal | Newly created collection. |

### 2.4.3.29 Property ConfigurationsWithErrors

```
// Get the collection of all configurations and
// set the errors interface to additionally receive
// configurations interface relevant errors.
[propget, id(28)]
HRESULT ConfigurationsWithErrors([out, retval]
                                    LPDISPATCH *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the collection of all configurations. |

### 2.4.3.30 Property `ExtendedCheck`

```
// Gets or sets the ExtendedCheck for a specific check
// category. Because extended checks could be very time
// consuming these checks have to have the possibility to be
// turned on and off.
[propget, id(31)]
HRESULT ExtendedCheck([in] BSTR CheckType,
                        [out, retval] BOOL *pVal);
[propput, id(31)]
HRESULT ExtendedCheck([in] BSTR CheckType, [in] BOOL newVal);
```

| Argument Name | Description |
|---|---|
| CheckType | The check category to get or set, e.g. GASSVXIChannels |
| *pVal | TRUE or FALSE. |

### 2.4.3.31 Method CategoryVersion

```
[propget, id(32)]
HRESULT CategoryVersion([in] BSTR Category ,[out, retval] float
*pVal);
```

| Argument Name | Description |
|---|---|
| Category | The Category as string, e.g. “Channels”, “Subsystems”, etc. |
| *pVal | The version number as float |

2.4.3.31.1 The CategoryVersion is used in ChannelEditor and SubsystemEditor to recognize already converted configurations.

### 2.4.3.32 Method GetTestCellID_ByName

```
[id(33)]
HRESULT GetTestCellID_ByName([in] BSTR TestCellName,[out,
retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| TestCellName | The name of the test cell |
| *pVal | The ID of the test cell. |

### 2.4.3.33 Property RTEUnchecked

```
[propget, id(34)]
HRESULT RTEUnchecked([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | A pointer to the RTE interface, but without doing a consistency check. |

2.4.3.33.1 The property RTEUnchecked results in the same object as using the property RTE, but there is no check for consistency. This can be used to retrieve an RTE object rapidly.

### 2.4.3.34 Method GetSensorCurve

```
[id(35)]
HRESULT GetSensorCurve([in] BSTR SensorName, [out] LPDISPATCH*
dataRecord);
```

| **Argument Name** | **Description** |
|---|---|
| SensorName | Name of the sensor to retrieve. First the sensor will be searched in database, after that in XML files |

| | |
|---|---|
| *dataRecord | Result pointer to a IDataRecord, which can be converted to IBreakpointTable2D, IBreakpointTable3D, IPolynomial. |

### 2.4.3.35 Method MergeSensorCurves

```
[id(36)]
HRESULT MergeSensorCurves([in] LPDISPATCH Sensor1, [in]
LPDISPATCH Sensor2, [out] LPDISPATCH* SensorMerged);
```

| Argument Name | Description |
|---|---|
| Sensor1, Sensor2 | Sensors to merge. Valid combinations are BreakpointTable2D – BreakpointTable2D, Polynomial – Polynomial and Polynomial – BreakpointTable2D. Polynomials must have degree 1, only when merging two polynomials the second one can have degree up to 4. |
| *SensorMerged | Result pointer to a IDataRecord, which can be converted to IBreakpointTable2D, IBreakpointTable3D, IPolynomial. |

### 2.4.3.36 Property SiteSpecificChannelAttributes

```
[propget, id(39)]
HRESULT SiteSpecificChannelAttributes([out, retval] LPDISPATCH
*pVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | The site specific channel attributes definition for the configuration data assigned to the *Application* interface. |

### 2.4.3.37 Property `TypeDescriptions`

```
[propget, id(40)]
HRESULT TypeDescriptions([out, retval] LPDISPATCH *pVal)
```

| Argument Name | Description |
|---|---|
| pVal | Information about available channel attributes |

### 2.4.4 Usage Conditions and Restrictions

2.4.4.1 The property *UseInactive* must be set before the collections of configuration data are accessed. If the property is changed after access to one of these collections their behaviour is undefined.

2.4.4.2 Valid values for *Category* in *IsLocked*, *DataRecordsCount*, *DataRecordsCountXML*, and *NewCollection* shall be *Subsystems*, *Channels*, *EngineeringUnits*, *Polynomials*, *BreakPointTables*, *TextOutputPages*, *TransientLogDefs*, *UserFunctions* and *Macros*. If *Category* has another value, an error shall be generated.

2.4.4.3 Valid values for *ConfigLevel* in *IsLocked* shall be 0 to 6 for the configuration levels as defined in 2.1.3.4 and -1 for all levels. Consequently, only the current configuration and its parent configurations can be locked.

2.4.4.4 When data are locked and the property is locked again an error shall be generated.

2.4.4.5 When data are locked and a client other than the one who has locked them requests a lock, an error shall be generated i.e. data can only be locked once.

2.4.4.6 When a client locks data and then deletes the connection without having unlocked the data, the data shall be unlocked automatically.

2.4.4.7 When more than one client is active, users are not actively informed about any change of data.

### 2.4.5 Persistent Data

2.4.5.1 The default configuration identification shall be stored in the configuration file.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.5 Interface "Configurations"

### 2.5.1 Description

2.5.1.1 This interface represents a collection of configurations. Single configurations are defined in 2.6.

### 2.5.2 Design

2.5.2.1 This interface shall be a dispatch interface.

2.5.2.2 This interface shall be an automation interface.

2.5.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *Configuration* interfaces in the collection. Deleting the default configuration shall not be allowed, i.e. an error shall be generated when this is tried.

2.5.2.4 This interface shall not be directly created. It shall be created by the *Configurations* property of the interface *Application*.

### 2.5.3 Methods and Properties

2.5.3.1 Property New

```
// Create a configuration.
[propget, id(1)]
HRESULT New([in] BSTR TestCell, [in] BSTR EngineType,
      [in] BSTR EngineStandard, [in] BSTR Customer,
      [in] long Id, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| TestCell | The test cell name. |
| EngineType | The engine type. |
| EngineStandard | The engine standard. |
| Customer | The customer name. |
| Id | The test configuration Identifier. |
| *pVal | The index of the new configuration. 0 if an error occurred. |

2.5.3.1.1 The properties of the newly created configuration shall be set to the values defined in the parameter list and all other values shall be set to default values.

2.5.3.1.2 The values specified in the parameter list will determine the Configuration Level of the new configuration.

2.5.3.1.3 The property NextConfigId (cf. 2.5.3.6) should be used to determine the next available configuration identifier when creating a Test Configuration.

### 2.5.3.2 Property Find

```
// Find a configuration.
[propget, id(2)]
HRESULT Find([in] LPDISPATCH Configuration,
      [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Configuration | A Configuration interface to be found. |
| *pVal | returns the index of the configuration, if it is found in the collection. If not, 0 shall be returned. |

### 2.5.3.3 Property FindById

```
// Find a test configuration by Id.
[propget, id(3)]
HRESULT FindById([in] long Id, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Id | A test configuration Id to be found. |
| *pVal | returns the index of the test configuration, if it is found in the collection. If not, 0 shall be returned. |

### 2.5.3.4 Property FindByAttributes

```
// Find a configuration in the collection using the attributes.
[propget, id(9)]
HRESULT FindByAttributes([in] BSTR TestCell,
[in] BSTR EngineType, [in] BSTR EngineStandard,
[in] BSTR Customer, [in] long Id, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| TestCell | The test cell name |
| EngineType | The engine type |
| EngineStandard | The engine standard |
| Customer | The customer name |
| Id | The test configuration identifier |
| *pVal | returns the index of the configuration, if it is found in the collection. If not, 0 shall be returned. |

### 2.5.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | denotes the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

#### 2.5.3.6 Property NextConfigId

```
// The Next Test Configuration Identifier
[propget, id(8] HRESULT NextConfigId([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | Returns the next configuration identifier. |

2.5.3.6.1 This property should be used by the clients of the Configuration Server to obtain the test configuration identifier to be passed as the argument "*Id*" of the property *New*.

#### 2.5.3.7 Method Refresh

```
[id(10] HRESULT Refresh();
```

2.5.3.7.1 The method *Refresh* reads the configurations again and adds new ones to the end of its list.

### 2.5.4 Usage Conditions and Restrictions

2.5.4.1 There must be at least one configuration available, the root configuration.

2.5.4.2 Configurations with child configurations may not be deleted. The root configuration may not be deleted. The configuration to be deleted will have to exist.

2.5.4.3 A configuration which is added must not already be in the collection. If a configuration is added an appropriate parent configuration must already exist in the collection.

### 2.5.5 Example

2.5.5.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interfaces *Configurations* and *Configuration.*

```
' Establish the connection to the server.
Set App = CreateObject ("prodas.Config")

' Get the collection of configurations.
Set Configs = App.Configurations

' Loop through the configurations.
for i = 1 to Configs.Count
   set Config = Configs (i)
   DisplayConfig Config
next

' Enumerate the configurations.
for each config in configs
   DisplayConfig config
next

' Search for configurations by Id.
id =  configs ((configs.Count+1)/2).Id
FindConfigId configs, id
FindConfigId configs, 123456789

' Create a new configuration and delete it again.
App.Identify "sl3", "sl3"
configIdx = Configs.New("", "AbsurdEngine", "", "", 0)
Set config = configs.Item(configIdx)
DisplayConfig config
configs.Delete(configIdx)

sub DisplayConfig (config)
   message = "test cell = " & config.TestCell & vbLf
   message = message & "engine type = " & config.EngineType &
vbLf
   message = message & "engine standard = " &
config.EngineStandard & vbLf
   message = message & "engine customer = " & config.Customer &
vbLf
   message = message & "description = " & config.Description &
vbLf
   message = message & "Id = " & config.Id & vbLf
   MsgBox message
end sub

sub FindConfigId (configs, id)
   set config = configs.FindById (id)
   if config = null then
      MsgBox "Configuration with Id " & id & " not found."
   else
      MsgBox "Configuration with Id " & id & " found."
      DisplayConfig (config)
   end if
end sub
```

## 2.6 Interface "Configuration"

### 2.6.1 Description

2.6.1.1 This interface represents a configuration, which may be a base configuration or a test configuration as defined in the Engineering Specification for Configuration Server.

### 2.6.2 Design

2.6.2.1 This interface shall be a dispatch interface.

2.6.2.2 This interface shall be an automation interface.

2.6.2.3 This interface shall not be directly created. It shall be created by the *CurrentConfiguration* or the *DefaultConfiguration* properties of the interface *Application* or by the *Item* or the *New* property of the interface *Configurations*.

### 2.6.3 Methods and Properties

2.6.3.1 Property TestCell

```
[propget, id(1)]
HRESULT TestCell([out, retval] BSTR *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the name of the test cell of the configuration. The default shall be an empty string, meaning that the test cell is not defined. |

2.6.3.2 Property EngineType

```
[propget, id(2)]
HRESULT EngineType([out, retval] BSTR *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the engine type of the configuration. The default shall be an empty string, meaning that the engine type is not defined. |

### 2.6.3.3 Property EngineStandard

```
[propget, id(3)]
HRESULT EngineStandard([out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the engine standard of the configuration. The default shall be an empty string, meaning that the engine standard is not defined. |

### 2.6.3.4 Property Customer

```
[propget, id(4)]
HRESULT Customer([out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the customer of the configuration. The default shall be an empty string, meaning that the customer is not defined. |

### 2.6.3.5 Property Description

```
// Description
[propget, id(7)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(7)]
HRESULT Description([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the comment associated with the configuration. The default shall be an empty string. |

### 2.6.3.6 Property Id

```
// Configuration Id
[propget, id(8)]
HRESULT Id([out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the identification of the test configuration only. It shall be provided when the test configuration is created by the *New* property of the *Configurations* interface. It shall be unique within the whole installation. |

### 2.6.3.7 Property ConfigLevel

```
// Configuration level
[propget, id(10)]
HRESULT ConfigLevel ([out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the configuration level value of the configuration according to 2.1.3.4. |

### 2.6.3.8 Property Users

```
// A collection of Users
[propget, id(11)]
HRESULT Users([out, retval] LPDISPATCH *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | a reference to the collection of users if the configuration is the root configuration, an engine type configuration or a test cell configuration. Otherwise an error shall be generated. |

2.6.3.8.1 Only these users will have write access to the configuration data at the root configuration level, the engine type configuration level or at the test cell configuration level respectively. The users with write access to one of these configuration levels will have write access to the associated child configuration data. All users have write access to the configuration data at the test configuration level.

### 2.6.3.9 Property IsEqual

```
// Comparison
[propget, id(12)]
HRESULT IsEqual([in] LPDISPATCH Configuration,
      [out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Configuration | A Configuration Interface. If *Configuration* does not allow accessing an *IConfiguration* interface an error shall be generated. |
| *pVal | indicates whether the configuration is equal to the configuration *Configuration.* |

### 2.6.3.10 Property LastModificationDate

```
// Last modification date
[propget, id(16)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.6.3.11 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed since the data have been written back to the configuration collection the last time. It shall initially be set to *false*. |

### 2.6.3.12 Method Import

```
// import the data contained in the file
[id(13)]
HRESULT Import([in] BSTR FileName);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |

2.6.3.12.1 This method imports the configuration description file.

2.6.3.12.2 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.6.3.13 Method Export

```
// Export the data into a file
[id(14)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(TRUE)] BOOL Force);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the configuration data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| Force | Flag to force an export of the configuration information, i.e. the *IsChanged* flag will be ignored. |

2.6.3.13.1 This method exports the configuration description file.

2.6.3.13.2 Note that the file name *FileName* shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computers.

### 2.6.3.14 Method ImportXML

```
// Import the data contained in an XML string.
[id(22)]
HRESULT ImportXML([in] VARIANT XML);
```

| **Argument Name** | **Description** |
|---|---|
| XML | Byte array containing the XML description of a configuration. |

2.6.3.14.1 This method imports the configuration description from an XML string.

### 2.6.3.15 Method ExportXML

```
// Export the data into an XML string.
[id(21)]
HRESULT ExportXML([out, retval] VARIANT *pXML);
```

| **Argument Name** | **Description** |
|---|---|
| *pXML | Byte array containing the XML description of a configuration. |

2.6.3.15.1 This method exports the configuration description to an XML string.

### 2.6.3.16 Method Save

```
// Save the configuration definition.
[id(23)]
HRESULT Save();
```

#### 2.6.3.17 Property DataRecordsCount

```
// Returns the number of data record contained in a
// configuration
[propget, id(18)]
HRESULT DataRecordsCount([in] BSTR Category,
      [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Category | The category associated to the configuration, i.e. BreakPointTables, Channels, EngineeringUnits, Macros, Polynomials, Subsystems, TextOutputPages, TransientLogDefs, or UserFunctions. |
| *pVal | The number of records of the category *Category* contained in the configuration. If the category is not valid, an error shall be generated. |

#### 2.6.3.18 Property IsAncestor

```
// Determine whether a configuration is a parent of this
// configuration
[propget, id(19)]
HRESULT IsAncestor ([in] LPDISPATCH Configuration,
      [out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Configuration | A Configuration Interface. If *Configuration* does not allow accessing an *IConfiguration* interface an error shall be generated. |
| *pVal | Indicates whether the configuration *Configuration* is a parent of this configuration. |

### 2.6.4 Usage Conditions and Restrictions

2.6.4.1 The properties *TestCell*, *EngineType*, *EngineStandard* and *Customer* must be usable as valid directory names.

### 2.6.5 Example

2.6.5.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interfaces *Configurations* and *Configuration.*

```
' Establish the connection to the server.
Set app = CreateObject ("prodas.Config")

' Get the current configuration which has been set to the
' default configuration.
Set config = app.CurrentConfiguration

' Set the current configuration for the connection.
' If the configuration does not exist, an error will be
generated.
app.CurrentConfiguration = config

' Check if the Id and the last modification date have been set
correctly.
MsgBox "Id = " & config.Id
MsgBox "Last Modification Date = " &
config.LastModificationDate
```

## 2.7 Interface "Errors"

### 2.7.1 Description

2.7.1.1 This interface represents the buffer for the error messages generated when checking configuration data.

### 2.7.2 Design

2.7.2.1 This interface shall be a dispatch interface.

2.7.2.2 This interface shall be an automation interface.

2.7.2.3 This interface shall not be directly created. It shall be created by the *Errors* property of the interface *Application*.

2.7.2.4 Note that this interface will **not** implement a collection interface.

### 2.7.3 Methods and Properties

2.7.3.1 Property IsEmpty (Obsolete)

```
[propget, id(1)]
HRESULT IsEmpty([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *Obsolete. TRUE*, if the message buffer is empty and *FALSE* otherwise. |

2.7.3.2 Property Message (Obsolete)

```
[propget, id(2)]
HRESULT Message([out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | Obsolete. The oldest message, if there is one, and an empty string otherwise. The returned message shall be removed from the buffer. |

2.7.3.3 Method Clear

```
[id(3)]
HRESULT Clear();
```

2.7.3.3.1 The method ***Clear*** shall clean the message buffer.

### 2.7.3.4 Method GetMessageItem

```
[id(4)]
HRESULT GetMessageItem([out] long* severity,[out] VARIANT*
time,[out, retval] BSTR *pVal);
```

| Argument Name | Description |
|---|---|
| *severity | Type of the message (0=Information, 1=Warnings, 2=Error, 3=FatalError). Currently, the Config Server only generates warnings and errors. Only errors can change the state of a configuration to inconsistent. |
| *time | Time when the event occured |
| *pVal | The message of the event. |

### 2.7.3.5 Property HasErrorsOrWarnings

```
[propget, id(5)]
HRESULT HasErrorsOrWarnings([out, retval] BOOL *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | TRUE if there are any messages, either warnings or errors. Otherwise FALSE. |

### 2.7.3.6 Property GetErrorOrWarning

```
[id(6)] HRESULT GetErrorOrWarning([out] VARIANT* severity,[out]
VARIANT* time,[out, retval] BSTR *pVal);
```

| Argument Name | Description |
|---|---|

| | |
|---|---|
| *severity | Type of the message<br><br>0 = Information<br>1=Warnings<br>2=Error<br>3=FatalError.<br><br>Currently, the Config Server only generates warnings and errors. Only errors can change the state of a configuration to inconsistent. |
| *time | Time when the event occured |
| *pVal | The message of the event. |

**NOTE** This method was  added for VBScript support, because VBScripts only support the data type VARIANT as the output parameter of a method.

### 2.7.4 Example (Obsolete)

2.7.4.1 The following is an example of a VBS client which illustrates how to use the obsolete properties and methods of the interface *Errors*.

```
'Establish the connection to the server.
Set App = CreateObject ("prodas.Config")

' Clear the error buffer.
Set Errors = App.Errors
Errors.Clear

' Check all data of the configuration.
IsConsistent = App.IsAllConsistent

' Display all errors to the user.
if Errors.IsEmpty then
   MsgBox "Configuration data consistent."
else
   count = 0
   do until Errors.IsEmpty
      count = count + 1
      MsgBox "Error " & count & ":" & Errors.Message
   loop
end if
```

2.7.4.2 The following is an example of a C# client which illustrates how to use the new properties and methods of the interface *Errors*.

```
'Establish the connection to the server.
```

```
CONFIGSERVERLib.IApplication app = new
CONFIGSERVERLib.ApplicationClass();

' Clear the error buffer.
CONFIGSERVERLib.IErrors errors = app.Errors;
Errors.Clear();

' Check all data of the configuration.
Int isConsistent = app.IsAllConsistent;

' Display all errors to the user.
int severity;
object time;

while (errors.HasErrorsOrWarnings == 1)
{
   string message = errors.GetMessageItem(out severity, out
time);
   if (severity == 0)
      MessageBox.Show(message, „Information“);
   else if (severity == 1)
      MessageBox.Show(message, „Warning“);
   else if (severity == 2)
      MessageBox.Show(message, „Error“);
   else if (severity == 3)
      MessageBox.Show(message, „FatalError“);
}
```

## 2.8 Interface "RTE"

### 2.8.1 Description

2.8.1.1 This interface generates the binary data formats required by the RTE and accessed by the Configuration Retriever.

### 2.8.2 Design

2.8.2.1 This interface shall be a dispatch interface.

2.8.2.2 This interface shall be an automation interface.

2.8.2.3 This interface shall not be directly created. It shall be created by the *RTE* property of the interface *Application*.

### 2.8.3 Methods and Properties

2.8.3.1 Property GenericChannelData

```
// Access generic channel data
[propget, id(1)]
HRESULT GenericChannelData ([in] long SubsystemId,
        [out, retval] VARIANT *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| SubsystemId | The Subsystem identifier. |
| *pVal | the generic channel data for the subsystem identified by *SubsystemId* required by the RTE in a one dimensional safe array. |

2.8.3.2 Property LimitsActionsData

```
[propget, id(2)]
HRESULT LimitsActionsData([out, retval] VARIANT *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the limits and actions data required by the RTE in a one dimensional safe array. |

### 2.8.3.3 Property ChannelData

```
// Access subsystem specific data
[propget, id(3)]
HRESULT ChannelData ([in] long SubsystemId,
      [out, retval] VARIANT *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| SubsystemId | The Subsystem identifier. |
| *pVal | the channel data for the subsystem identified by *SubsystemId* required by the RTE in a one dimensional safe array. |

### 2.8.3.4 Property LookupTables

```
// Access all Lookup Tables
[propget, id(4)]
HRESULT LookupTables([out, retval] VARIANT *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | all the lookup table data required by the RTE in a one dimensional safe array. |

### 2.8.3.5 Property TestConfiguration

```
// Access Test Configuration information
[propget, id(5)]
HRESULT TestConfiguration([in] long TestId, [out, retval]
VARIANT *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| TestId | Test identification used to retrieve data from the test result database. |
| *pVal | the test configuration data required by the RTE in a one dimensional safe array. |

2.8.3.6 Property FileData

```
[propget, id(6)]
HRESULT FileData([in] long SubsystemId, [out, retval] VARIANT
*pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the file data required by the RTE in a one dimensional safe array. |

2.8.3.7 Method GetSensorCurve

```
[id(7)]
HRESULT GetSensorCurve([in] BSTR SensorName, [out] LPDISPATCH*
dataRecord);
```

| **Argument Name** | **Description** |
|---|---|
| SensorName | Name of the sensor to retrieve. First the sensor will be searched in database, after that in XML files |
| *dataRecord | Result pointer to a IDataRecord, which can be converted to IBreakpointTable2D, IBreakpointTable3D, IPolynomial. |

2.8.3.8 Method MergeSensorCurves

```
[id(8)]
HRESULT MergeSensorCurves([in] LPDISPATCH Sensor1, [in]
LPDISPATCH Sensor2, [out] LPDISPATCH* SensorMerged);
```

| **Argument Name** | **Description** |
|---|---|
| Sensor1, Sensor2 | Sensors to merge. Valid combinations are BreakpointTable2D – BreakpointTable2D, Polynomial – Polynomial and Polynomial – BreakpointTable2D. Polynomials must have degree 1, only when merging two polynomials the second one can have degree up to 4. |
| *SensorMerged | Result pointer to a IDataRecord, which can be converted to IBreakpointTable2D, IBreakpointTable3D, IPolynomial. |

### 2.8.4 Usage Conditions and Restrictions

2.8.4.1 In all properties, the return values must be a one dimensional safe array.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

2.8.4.2 *SubsystemId* and *TestId* must be valid.

## 2.9 Interface "Subsystems"

### 2.9.1 Description

2.9.1.1 This interface shall define the specific subsystems implemented. These are divided into Calculated, External and Hardware subsystems.

### 2.9.2 Design

2.9.2.1 This interface shall be a dispatch interface.

2.9.2.2 This interface shall be an automation interface.

2.9.2.3 This interface shall not be directly created. It shall be created by the *Subsystems* property of the interface *Application*.

### 2.9.3 Methods and Properties

2.9.3.1 Property New

```
// Create a data record.
[propget, id(1)]
HRESULT New([in] BSTR Type, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Type | The subsystem type to be created. |
| *pVal | The index of the newly created subsystem in the collection. |

2.9.3.1.1 Note that it will not only create an interface *Subsystem*, but eventually one of the interfaces derived from it. The properties shall be set to the default values indicated in the respective section of subsystems.

2.9.3.1.2 The valid value for the *Type* parameter and the created interface shall be according to the Interface Control Document for the Configuration Server-Subsystems. If the *Type* parameter has any other value, an error shall be generated.

### 2.9.3.2 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | The name of the subsystem to be found. |
| *pVal | the index of the subsystem with name *Name*, if it is found in the collection, and 0 otherwise. |

### 2.9.3.3 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

### 2.9.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.9.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.9.3.6 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the subsystem data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.9.3.6.1 Note that the file name *FileName* shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computers.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.9.3.7 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.9.3.7.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.9.3.8 Property FindById

```
// Find a Subsystem by Identifier.
[propget, id(2)]
HRESULT FindById([in] long Id, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Id | The Identifier of the subsystem to be found. |
| *pVal | the index of the subsystem with identifier *Id*, if it is found in the collection, and 0 otherwise. |

## 2.9.4 Usage Conditions and Restrictions

2.9.4.1 The *Type* parameter in *New* must have a valid value.

## 2.10 Interface "Subsystem"

### 2.10.1 Description

2.10.1.1 This interface represents a generic subsystem.

### 2.10.2 Design

2.10.2.1 This interface shall be a dispatch interface.

2.10.2.2 This interface shall be an automation interface.

2.10.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *Subsystems*.

2.10.2.4 This interface will be aggregated to the more complex subsystem interfaces.

### 2.10.3 Methods and Properties

2.10.3.1 Property Id

```
// Subsystem Identifier
[propget, id(113)]
HRESULT Id([out, retval] long *pVal);
[propput, id(113)]
HRESULT Id([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Id of the subsystem for the real-time data engine. The default value shall be -1. If the value is outside the range [0-29] or it is set to the Id of another subsystem of the same configuration an error shall be generated. |

2.10.3.2 Property Type

```
// Subsystem Type
[propget, id(114)]
HRESULT Type([out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the type of a subsystem, as described in 2.9. |

### 2.10.3.3 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed since the data have been written back to the subsystem collection the last time. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.10.3.4 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.10.3.5 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

### 2.10.3.6 Property ConfigLevel

```
[propget, id(3002)]
HRESULT ConfigLevel ([out, retval] long *pVal);
[propput, id(3002)]
HRESULT ConfigLevel ([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration level to which the data record belongs according to 2.1.3.4. If the property is set to a value which is not between 0 and this *ConfigLevel* value an error shall be generated. |

### 2.10.3.7 Property Name

```
// The Data Record Name
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the data record. It shall uniquely identify the subsystem. The default value shall be an empty string. If an empty string is set or it is set to the name of another data record of the same configuration an error shall be generated. |

#### 2.10.3.8 Property Description

```
// Description
[propget, id(3005)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the description of the subsystem. The default value shall be an empty string. |

#### 2.10.3.9 Property CreationDate

```
[propget, id(115)]
HRESULT CreationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | The creation date and time of the subsystem. |

### 2.10.4 Usage Conditions and Restrictions

2.10.4.1 Properties which are specific for one or more subsystem types will not have any effect on other subsystem types.

## 2.11 Interface "Channels"

### 2.11.1 Description

2.11.1.1 This interface represents a collection of channels. A single channel is defined in 2.12

### 2.11.2 Design

2.11.2.1 This interface shall be a dispatch interface.

2.11.2.2 This interface shall be an automation interface.

2.11.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *Channel* interfaces in the collection.

2.11.2.4 This interface shall not be directly created. It shall be created by the *Channels* property of the interface *Application*.

### 2.11.3 Methods and Properties

2.11.3.1 Property New

```
// Create a data record.
[propget, id(1)]
HRESULT New([in] BSTR SubsystemName, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| SubsystemName | The *SubsystemName* must be a name that has already been defined via the Subsystems interface. If the name has not been defined, an error shall be generated |
| *pVal | The index of the newly created channel in the collection. |

### 2.11.3.2 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | The name of the channel to be found. |
| *pVal | the index of the data record with name *Name*, if it is found in the collection, and 0 otherwise. |

### 2.11.3.3 Method ImportTabDelimited

```
// Import tab delimited files
[id(2)]
HRESULT ImportTabDelimited([in] BSTR FileName,
      [in] BSTR Subsystem,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue(FALSE)] BOOL Replace);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the syntax is incorrect an error shall be generated. |
| Subsystem | The name of the subsystem to be imported. It must be a name that has already been defined via the Subsystems interface. If the name has not been defined, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Replace | The *Replace* flag, with default *false*, shall indicate whether the existing data are replaced or the new data are added. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

2.11.3.3.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

2.11.3.4 Method ExportTabDelimited

```
// Export tab delimited files
[id(3)]
HRESULT ExportTabDelimited([in] BSTR FileName,
      [in] BSTR Subsystem,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the channel data of the current configuration and its parent configurations into the file named by *FileName* in tab delimited format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| Subsystem | The name of the subsystem to be exported. It must be a name that has already been defined via the Subsystems interface. If the name has not been defined, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.11.3.4.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.11.3.5 Method ImportTabDelimitedText

```
// Import data from a string in tab delimited format.
[id(4)]
HRESULT ImportTabDelimitedText([in] VARIANT Text,
[in] BSTR Subsystem,
[in, optional, defaultvalue(-1)] long ConfigLevel,
[in, optional, defaultvalue(FALSE)] BOOL Replace);
```

| **Argument Name** | **Description** |
|---|---|
| Text | Byte array containing the channel description in tab delimited format. |
| Subsystem | The name of the subsystem to be imported. It must be a name that has already been defined via the Subsystems interface. If the name has not been defined, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Replace | The *Replace* flag, with default *false*, shall indicate whether the existing data are replaced or the new data are added. |

### 2.11.3.6 Method ExportTabDelimitedText

```
// Export data to a string in tab delimited format.
[id(5)]
HRESULT ExportTabDelimitedText([out] VARIANT* pText,
[in] BSTR Subsystem,
[in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| *pText | Byte array containing the channel description in tab delimited format. |
| Subsystem | The name of the subsystem to be exported. It must be a name that has already been defined via the Subsystems interface. If the name has not been defined, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

### 2.11.3.7 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

### 2.11.3.8 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.11.3.9 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

#### 2.11.3.10 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the channel data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.11.3.10.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

#### 2.11.3.11 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.11.3.11.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.12 Interface "Channel"

### 2.12.1 Description

2.12.1.1 This interface shall represent a channel. All acquired data, whether measured or calculated, are produced by a channel.

### 2.12.2 Design

2.12.2.1 This interface shall be a dispatch interface.

2.12.2.2 This interface shall be an automation interface.

2.12.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *Channels*.

### 2.12.3 Methods and Properties

2.12.3.1 Property SubsystemName

```
// Subsystem name of the channel
[propget, id(102)]
HRESULT SubsystemName([out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the subsystem to which the channel belongs. |

2.12.3.2 Property SubsystemType

```
// Subsystem type of the channel
[propget, id(103)]
HRESULT SubsystemType([out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the type of the subsystem defined by the property *SubsystemType* (cf. 2.9.3.1.2). |

### 2.12.3.3 Property AlternateName

```
[propget, id(109)]
HRESULT AlternateName([out, retval] BSTR *pVal);
[propput, id(109)]
HRESULT AlternateName([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the alternate name of the channel, used for display purposes. The default value shall be an empty string. |

### 2.12.3.4 Property DataType

```
[propget, id(110)]
HRESULT DataType([out, retval] BSTR *pVal);
[propput, id(110)]
HRESULT DataType([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the channel type, which shall be one of *Float*, *Boolean*, *Integer*, *Date*, or *Time*. If it is set to another value an error shall be generated. When the type is set the *Format* property and the *InitialValue* property shall be set to the default value of the type. |

### 2.12.3.5 Property InitialValue

```
[propget, id(111)]
HRESULT InitialValue([out, retval] float *pVal);
[propput, id(111)]
HRESULT InitialValue([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the initial value of an output channel, which shall also serve as the substitute value of an output channel when the reference value is *BAD*. If the property is accessed for a channel which is not an output channel an error shall be generated. The default value shall be 0.0. Note that this property may also be used for channels which are not float channels. |

### 2.12.3.6 Property Format

```
[propget, id(112)]
HRESULT Format([out, retval] BSTR *pVal);
[propput, id(112)]
HRESULT Format([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the default format for display of the channel's values. The valid values will depend on the type of the channel, but they will not be checked. The default value shall be *%f* for type *Float*, %d for type *Integer*, *false/true* for type *Boolean*, *%Y-%m-%d* for type *Date* and *%H:%M:%S* for type *Time*. |

### 2.12.3.7 Property UpdateRate

```
[propget, id(113)]
HRESULT UpdateRate([out, retval] long *pVal);
[propput, id(113)]
HRESULT UpdateRate([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the update rate of the channel in Hz. The default value shall be 1. |

### 2.12.3.8 Property QualityCeiling

```
[propget, id(114)]
HRESULT QualityCeiling([out, retval] BSTR *pVal);
[propput, id(114)]
HRESULT QualityCeiling([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the maximum quality of the channel's values. It shall have one of the values *GOOD*, *SUSPECT*, or *BAD*. The default value shall be *GOOD*. |

### 2.12.3.9 Property IsCritical

```
[propget, id(115)]
HRESULT IsCritical([out, retval] BOOL *pVal);
[propput, id(115)]
HRESULT IsCritical([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the channel is a critical one. This will especially mean that it will be logged as part of the critical log. The default value shall be *false*. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.12.3.10 Property EngineeringUnit

```
[propget, id(116)]
HRESULT EngineeringUnit([out, retval] BSTR *pVal);
[propput, id(116)]
HRESULT EngineeringUnit([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the engineering unit of the channel. The default value shall be an empty string, which shall denote that the engineering unit is undefined. If the property is set to a non-empty string which is not defined in the list of engineering units (cf. 2.17) an error shall be generated. The property will be ignored for Boolean channels. |

### 2.12.3.11 Property DisplayMin

```
[propget, id(117)]
HRESULT DisplayMin([out, retval] float *pVal);
[propput, id(117)]
HRESULT DisplayMin([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the minimum value used in diverse graphical displays. The default value shall be 0.0. The property will be ignored for Boolean channels. |

### 2.12.3.12 Property DisplayMax

```
[propget, id(118)]
HRESULT DisplayMax([out, retval] float *pVal);
[propput, id(118)]
HRESULT DisplayMax([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the maximum value used in diverse graphical displays. The default value shall be 1.0. The property will be ignored for Boolean channels. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.12.3.13 Property IsOutput

```
[propget, id(119)]
HRESULT IsOutput([out, retval] BOOL *pVal);
[propput, id(119)]
HRESULT IsOutput([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the channel is an output channel. The default value shall be *false*. |

### 2.12.3.14 Property ReferenceChannel

```
[propget, id(120)]
HRESULT ReferenceChannel([out, retval] BSTR *pVal);
[propput, id(120)]
HRESULT ReferenceChannel([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the reference channel for an output channel. If the property is accessed for a channel which is not an output channel an error shall be generated. |

### 2.12.3.15 Property Group

```
[propget, id(122)]
HRESULT Group([out, retval] BSTR *pVal);
[propput, id(122)]
HRESULT Group([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the group the channel belongs to. The default value shall be an empty string. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

#### 2.12.3.16 Property DeadBand

```
[propget, id(123)]
HRESULT DeadBand([out, retval] float *pVal);
[propput, id(123)]
HRESULT DeadBand([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the value of the deadband used for reporting an alarm state transition. The default value shall be 0.0. The property will be ignored for Boolean channels. |

#### 2.12.3.17 Property AlarmLimits

```
// Access to alarm monitoring information
[propget, id(124)]
HRESULT AlarmLimits([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | returns a reference to the collection of alarm limit definitions (cf. 2.13). |

#### 2.12.3.18 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.12.3.19 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.12.3.20 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

### 2.12.3.21 Property ConfigLevel

```
// Configuration level of the data record
[propget, id(3002)]
HRESULT ConfigLevel([out, retval] long *pVal);
[propput, id(3002)]
HRESULT ConfigLevel([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration level to which the data record belongs according to 2.1.3.4. If the property is set to a value which is not between 0 and this *ConfigLevel* value an error shall be generated. |

### 2.12.3.22 Property Name

```
// Name of the data record
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the channel. It shall uniquely identify the channel. The default value shall be an empty string. If an empty string is set or the name is set to the name of another channel within the same configuration level an error shall be generated.<br><br>The maximum length is 39 characters. |

### 2.12.3.23 Property Description

```
[propget, id(3005)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the description of the channel. The default value shall be an empty string. |

### 2.12.3.24 Property SignalId

```
[propget, id(129)]
HRESULT SignalId([out, retval] BSTR *pVal);
[propput, id(129)]
HRESULT SignalId([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | The identification of the signal path. |

### 2.12.3.25 Property PinoutDescription

```
[propget, id(130)]
HRESULT PinoutDescription([out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | If channel subsystem has an PBSPort, DPSPort, TSMModule or GASSPinout the appropriate description of the port, module or pinout is returned, otherwise “” is returned. |

### 2.12.3.26 Property CalSensor

```
[propget, id(131)]
HRESULT CalSensor([out, retval] BSTR *pVal);
[propput, id(131)]
HRESULT CalSensor([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | The calibration sensor for the channel. The default value shall be an empty string. |

### 2.12.3.27 Property CalGroup

```
[propget, id(132)]
HRESULT CalGroup([out, retval] BSTR *pVal);
[propput, id(132)]
HRESULT CalGroup([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | The calibration group for the channel. The default value shall be an empty string. |

### 2.12.3.28 Property FixedSensor

```
[propget, id(133)]
HRESULT FixedSensor([out, retval] BSTR *pVal);
[propput, id(133)]
HRESULT FixedSensor([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | The fixed sensor for the channel. The default value shall be an empty string. |

### 2.12.3.29 Property CustomerSensor

```
[propget, id(135)]
HRESULT CustomerSensor ([out, retval] BSTR *pVal);
[propput, id(135)]
HRESULT CustomerSensor ([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | The customer sensor for the channel. The default value shall be an empty string. |

### 2.12.3.30 Property SiteSpecificAttribute

```
[propput, id(134)
HRESULT SiteSpecificAttribute(
[in] BSTR Name,
[in] VARIANT newVal);

[propget, id(134)
HRESULT SiteSpecificAttribute(
[in] BSTR Name,
[out, retval] VARIANT *pVal);
```

| Argument Name | Description |
|---|---|
| Name | The name of the attribute |
| *pVal | Returns a VARIANT containing the value of the attribute. |

## 2.13 Interface "AlarmLimits"

### 2.13.1 Description

2.13.1.1 This interface represents the collection of alarm definitions of a channel. A single alarm definition is defined in 2.14.

### 2.13.2 Design

2.13.2.1 This interface shall be a dispatch interface.

2.13.2.2 This interface shall be an automation interface.

2.13.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *AlarmLimit* interfaces in the collection.

2.13.2.4 This interface shall not be directly created. It shall be created by the *AlarmLimits* property of the interface *Channel*.

### 2.13.3 Methods and Properties

2.13.3.1 Property New

```
// Create a data record.
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | The index of the newly created *AlarmLimit*. |

2.13.3.2 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.13.4 Usage Conditions and Restrictions

2.13.4.1 Only one alarm definition of each type may be present. Thus the number of alarm definitions will be restricted to 3 for Boolean channels and to 5 for the other channel types.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.14 Interface "AlarmLimit"

### 2.14.1 Description

2.14.1.1 This interface represents an alarm definition of a channel.

### 2.14.2 Design

2.14.2.1 This interface shall be a dispatch interface.

2.14.2.2 This interface shall be an automation interface.

2.14.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *AlarmLimits*.

### 2.14.3 Methods and Properties

2.14.3.1 Property Type

```
// Type of the alarm
[propget, id(1)]
HRESULT Type([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT Type([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the type of the alarm. The valid types shall be *LOLO*, *LO*, *HI*, *HIHI* and *ROC* for *Float* channels and *DISCINFO*, *DISCALARM* and *DISCQUIET* for *Boolean* channels. The meaning of these types is defined in the Engineering Specification for Configuration Server. |

2.14.3.2 Property LimitIsConstant

```
[propget, id(4)]
HRESULT LimitIsConstant([out, retval] BOOL *pVal);
[propput, id(4)]
HRESULT LimitIsConstant([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the limit of the alarm is constant, i.e. if the *LimitValue* is valid or the *LimitChannel*. The default value shall be *true*. The property will be ignored for Boolean channels. |

### 2.14.3.3 Property LimitValue

```
[propget, id(5)]
HRESULT LimitValue([out, retval] float *pVal);
[propput, id(5)]
HRESULT LimitValue([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the constant limit value for the alarm. It will only be valid if *LimitIsConstant* is *true*. The default value shall be 0.0. |

### 2.14.3.4 Property LimitChannel

```
[propget, id(6)]
HRESULT LimitChannel([out, retval] BSTR *pVal);
[propput, id(6)]
HRESULT LimitChannel([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the limit channel. It will only be valid if *LimitIsConstant* is *false*. The default value shall be an empty string. If it is set to a value which is not the name of a channel an error shall be generated. |

### 2.14.3.5 Property DelayTime

```
[propget, id(7)]
HRESULT DelayTime([out, retval] float *pVal);
[propput, id(7)]
HRESULT DelayTime([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the delay time until an alarm is triggered in seconds. The default value shall be 0. The property will be ignored for Boolean channels. |

### 2.14.3.6 Property Actions

```
// Access to the collection of related actions.
[propget, id(9)]
HRESULT Actions([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | returns a reference to the associated actions collection. |

### 2.14.3.7 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

## 2.15 Interface "Actions"

### 2.15.1 Description

2.15.1.1 This interface represents the collection of actions of an alarm limit definition. A single action is defined in 2.16.

### 2.15.2 Design

2.15.2.1 This interface shall be a dispatch interface.

2.15.2.2 This interface shall be an automation interface.

2.15.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *Action* interfaces in the collection.

2.15.2.4 This interface shall not be directly created. It shall be created by the *Actions* property of the interface *AlarmLimit.*

### 2.15.3 Methods and Properties

2.15.3.1 Property New

```
// Create a data record.
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | creates a new action and return its index. |

2.15.3.2 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.15.4 Usage Conditions and Restrictions

2.15.4.1 At most 10 actions may be created per alarm limit definition, 5 actions for channel value increasing and 5 actions for channel value decreasing when crossing the alarm limit.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.16 Interface "Action"

### 2.16.1 Description

2.16.1.1 This interface represents an action of an alarm limit definition.

### 2.16.2 Design

2.16.2.1 This interface shall be a dispatch interface.

2.16.2.2 This interface shall be an automation interface.

2.16.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *Actions*.

### 2.16.3 Methods and Properties

2.16.3.1 Property Type

```
// Action type
[propget, id(1)]
HRESULT Type([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT Type(BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the type of the action. The valid types shall be *StartUserLog*, *StopUserLog*, *SetChannel*, *ResetChannel*, *EventComment*, *Fullset*, *InvokeScript*, *SetQualityGood*, *SetQualityBad*, *SetQualitySuspect* or *SaveCriticalLog*. |

2.16.3.2 Property LogFile

```
[propget, id(2)]
HRESULT LogFile([out, retval] BSTR *pVal);
[propput, id(2)]
HRESULT LogFile([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the transient log file for the action types *StartUserLog* and *StopUserLog*. If it is accessed for other types an error shall be generated. |

### 2.16.3.3 Property ChannelName

```
[propget, id(3)]
HRESULT ChannelName([out, retval] BSTR *pVal);
[propput, id(3)]
HRESULT ChannelName([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of a channel. This property must be set if the action type is one of *SetChannel*, *ResetChannel*, *SetQualityGood*, *SetQualitySuspect* or *SetQualityBad*. If the channel name does not exist, or if the action type is set to *SetChannel* or *ResetChannel* and the channel type is not Boolean, an error shall be generated. |

### 2.16.3.4 Property ScriptFile

```
[propget, id(4)]
HRESULT ScriptFile([out, retval] BSTR *pVal);
[propput, id(4)]
HRESULT ScriptFile([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the script file for the action type *InvokeScript*. If it is accessed for another type an error shall be generated. It shall not be checked whether the script file exists. |

### 2.16.3.5 Property Text

```
[propget, id(5)]
HRESULT Text([out, retval] BSTR *pVal);
[propput, id(5)]
HRESULT Text([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the event text for the action type *EventComment*. If it is accessed for other types an error shall be generated. |

### 2.16.3.6 Property AveragingTime

```
[propget, id(8)]
HRESULT AveragingTime([out, retval] float *pVal);
[propput, id(8)]
HRESULT AveragingTime([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the averaging time for the action type *Fullset* in seconds. If it is accessed for another type an error shall be generated. |

### 2.16.3.7 Property Priority

```
[propget, id(9)]
HRESULT Priority([out, retval] BSTR *pVal);
[propput, id(9)]
HRESULT Priority ([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the priority for the action. This can be Low, Medium or High. The default shall be High. |

### 2.16.3.8 Property IsTriggerPositive

```
[propget, id(10)]
HRESULT IsTriggerPositive([out, retval] BOOL *pVal);
[propput, id(10)]
HRESULT IsTriggerPositive([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the alarm triggers the action when the limit is reached while the channel value is increasing (*true*) or decreasing. It shall initially be set to *true*. |

### 2.16.3.9 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. |

## 2.16.4 Usage Conditions and Restrictions

2.16.4.1 The properties *LogFile*, *ChannelName*, *ScriptFile*, *Text* and *AveragingTime* may only be accessed for the appropriate action types as defined above.

## 2.17 Interface "EngineeringUnits"

### 2.17.1 Description

2.17.1.1 This interface represents a collection of engineering units. A single engineering unit is defined in 2.18.

### 2.17.2 Design

2.17.2.1 This interface shall be a dispatch interface.

2.17.2.2 This interface shall be an automation interface.

2.17.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *EngineeringUnit* interfaces in the collection.

2.17.2.4 This interface shall not be directly created. It shall be created by the *EngineeringUnits* property of the interface *Application*.

### 2.17.3 Methods and Properties

2.17.3.1 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | the engineering unit with name *Name*. |
| *pVal | the index of the data record with name *Name*, if it is found in the collection, and 0 otherwise. |

2.17.3.2 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

### 2.17.3.3 Property New

```
// Create a data record.
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | The index of the newly created engineering unit in the collection. |

### 2.17.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.17.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.17.3.6 Method Export

```
// Export the data into a file
[id(2)]
HRESULT Export([in] BSTR FileName);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the Engineering Units data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |

2.17.3.6.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.17.3.7 Method Import

```
// import the data contained in the file
[id(1)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.17.3.7.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.17.3.8 Method GetPressureCategory

[id(3), helpstring("method GetPressureCategory")] HRESULT GetPressureCategory([out, retval] BSTR *pVal);

| **Argument Name** | **Description** |
|---|---|
| *pVal | The name of the pressure engineering units category. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

#### 2.17.3.9 Method GetTemperatureCategory

[id(4), helpstring("method GetTemperatureCategory")] HRESULT GetTemperatureCategory([out, retval] BSTR *pVal);

| **Argument Name** | **Description** |
|---|---|
| *pVal | The name of the temperature engineering units category. |

## 2.18 Interface "EngineeringUnit"

### 2.18.1 Description

2.18.1.1 This interface represents an engineering unit, which is used to interpret the value of a float channel.

### 2.18.2 Design

2.18.2.1 This interface shall be a dispatch interface.

2.18.2.2 This interface shall be an automation interface.

2.18.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *EngineeringUnits*.

### 2.18.3 Methods and Properties

2.18.3.1 Property Name

```
// Name of the data record
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the engineering unit. The default value shall be an empty string. An error shall be generated if the name is set to an empty string. |

2.18.3.2 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

### 2.18.3.3 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.18.3.4 Property Category

```
[propget, id(4)]
HRESULT Category ([out, retval] BSTR *pVal);
[propput, id(4)]
HRESULT Category ([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the category of the engineering unit. A category of engineering units will be a set which can be converted from one to the other by linear transformations. The default value shall be an empty string, indicating that the engineering unit does not belong to any category. |

### 2.18.3.5 Property IsPrimary

```
[propget, id(5)]
HRESULT IsPrimary([out, retval] BOOL *pVal);
[propput, id(5)]
HRESULT IsPrimary([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the engineering unit is the primary one of its category. The default value shall be *false*. |

### 2.18.3.6 Property Gain

```
[propget, id(6)]
HRESULT Gain ([out, retval] float *pVal);
[propput, id(6)]
HRESULT Gain ([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the factor used for the linear transformation of the primary unit of the engineering unit's category into the engineering unit. The default value shall be 1.0. |

### 2.18.3.7 Property Offset

```
[propget, id(7)]
HRESULT Offset([out, retval] float *pVal);
[propput, id(7)]
HRESULT Offset([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the offset used for the linear transformation of the primary unit of the engineering unit's category into the engineering unit. The default value shall be 0.0. |

### 2.18.3.8 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.18.4 Usage Conditions and Restrictions

2.18.4.1 The name will have to be a non-empty string with no white space or control characters.

2.18.4.2 The Engineering Units will be defined in the *Root* directory and shall be valid for all configurations.

### 2.18.5 Example

2.18.5.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interfaces *EngineeringUnits* and *EngineeringUnit*.

```
'Establish the connection to the server.
Set App = CreateObject ("prodas.Config")

Set Units = App.EngineeringUnits

Count = Units.Count

Index = Units.New
Set Unit1 = Units(Index)
Unit1.Name = "ExampleUnit1"
Unit1.Category = "Test"
Unit1.IsPrimary = true

Index = Units.New
Set Unit2 = Units(Index)
Unit2.Name = "ExampleUnit2"
Unit2.Category = "Test"
Unit2.Offset = 273.15

MsgBox "Created " & Units.Count - Count & " new Engineering
Units."
if Units.IsConsistent then
MsgBox "Engineering Units are consistent."
else
MsgBox "Engineering Units are inconsistent."
end if
```

## 2.19 Interface "Polynomials"

### 2.19.1 Description

2.19.1.1 This interface represents a collection of polynomials. Single polynomials are defined in 2.20.

### 2.19.2 Design

2.19.2.1 This interface shall be a dispatch interface.

2.19.2.2 This interface shall be an automation interface.

2.19.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *Polynomial* interfaces in the collection.

2.19.2.4 This interface shall not be directly created. It shall be created by the *Polynomials* property of the interface *Application*.

### 2.19.3 Methods and Properties

2.19.3.1 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | the Polynomial with name *Name*. |
| *pVal | the index of the data record with name *Name*, if it is found in the collection, and 0 otherwise. |

2.19.3.2 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

#### 2.19.3.3 Property New

```
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | The index of the newly created Polynomial in the collection. |

#### 2.19.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

#### 2.19.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.19.3.6 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the polynomial data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.19.3.6.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

## 2.19.3.7 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.19.3.7.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.20 Interface "Polynomial"

### 2.20.1 Description

2.20.1.1 This interface represents a polynomial defined on a finite x interval.

### 2.20.2 Design

2.20.2.1 This interface shall be a dispatch interface.

2.20.2.2 This interface shall be an automation interface.

2.20.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *Polynomials*.

### 2.20.3 Methods and Properties

2.20.3.1 Property Name

```
// Name of the data record
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the polynomial. When set, it must be non-empty and must only contain letters, digits, dot and underscores, otherwise an error shall be generated. |

2.20.3.2 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

#### 2.20.3.3 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

#### 2.20.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.20.3.5 Property ConfigLevel

```
// Configuration level of the polynomial
[propget, id(3002)]
HRESULT ConfigLevel([out, retval] long *pVal);
[propput, id(3002)]
HRESULT ConfigLevel([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration level to which the data record belongs according to 2.1.3.4. The default value shall be the *ConfigLevel* value of the current configuration. If the property is set to a value which is not between 0 and this *ConfigLevel* value an error shall be generated. |

### 2.20.3.6 Property Degree

```
// Degree of the polynomial
[propget, id(6)]
HRESULT Degree([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the degree of the polynomial, i.e. the highest index of a non-zero coefficient. |

### 2.20.3.7 Property Description

```
[propget, id(3005)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the description of the polynomial. The default value shall be an empty string. |

#### 2.20.3.8 Property XEngineeringUnit

```
[propget, id(8)]
HRESULT XEngineeringUnit ([out, retval] BSTR *pVal);
[propput, id(8)]
HRESULT XEngineeringUnit ([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the engineering unit of the x values of the polynomial. The default value shall be an empty string. When set, the engineering unit must be present in the engineering units' collection, otherwise an error shall be generated. |

#### 2.20.3.9 Property YEngineeringUnit

```
[propget, id(9)]
HRESULT YEngineeringUnit ([out, retval] BSTR *pVal);
[propput, id(9)]
HRESULT YEngineeringUnit ([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the engineering unit of the y values of the polynomial. The default value shall be an empty string. When set, the engineering unit must be present in the engineering units' collection, otherwise an error shall be generated. |

#### 2.20.3.10 Property Coefficient

```
[propget, id(10)]
HRESULT Coefficient ([in] long Index, [out, retval] float
*pVal);
[propput, id(10)]
HRESULT Coefficient ([in] long Index, [in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Index | If *Index* is smaller than 0, an error shall be generated. Note that *Degree* will be incremented if *Index* is greater than the current *Degree*, and that *Degree* will be decremented if the non-zero coefficient with the highest index is set to 0 |
| *pVal, newVal | the *Index*-th polynomial coefficient. |

### 2.20.3.11 Property Min

```
[propget, id(11)]
HRESULT Min([out, retval] float *pVal);
[propput, id(11)]
HRESULT Min([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the minimum value of the domain where the polynomial is defined. The default value shall be -3.402823466e+38. |

### 2.20.3.12 Property Max

```
[propget, id(12)]
HRESULT Max([out, retval] float *pVal);
[propput, id(12)]
HRESULT Max([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the maximum  value of the domain where the polynomial is defined. The default value shall be 3.402823466e+38. |

### 2.20.3.13 Property Value

```
// Evaluation of the polynomial
[propget, id(13)]
HRESULT Value([in] float x, [out, retval] float *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| x | the x coordinate value to be evaluated in the polynomial. An error shall be generated if *Min* <= *x* <= *Max* does not hold. |
| *pVal | the value of the polynomial for the argument *x*. |

### 2.20.4 Usage Conditions and Restrictions

2.20.4.1 The argument *Index* in the Coefficient property will have to be 0 or positive.

### 2.20.5 Example

2.20.5.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interfaces *Polynomials* and *Polynomial*.

```
' Establish the connection to the server.
Set app = CreateObject ("proDAS.Config")
Set polys = app.Polynomials

Index = polys.New
Set poly = polys (Index)
poly.Coefficient (0) = 1
poly.Coefficient (1) = -1
poly.Coefficient (2) = 1
poly.Min = 0
poly.Max = 1000

x = 20
MsgBox "Value at " & x & " is " & poly.Value (x)
```

## 2.21 Interface "BreakPointTables"

### 2.21.1 Description

2.21.1.1 This interface represents a collection of break point tables. Single break point tables may be 2 dimensional (cf. 2.22) or 3 dimensional (cf. 2.23).

### 2.21.2 Design

2.21.2.1 This interface shall be a dispatch interface.

2.21.2.2 This interface shall be an automation interface.

2.21.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *BreakPointTable2d* and *BreakPointTable3d* interfaces in the collection.

2.21.2.4 This interface shall not be directly created. It shall be created by the *BreakPointTables* property of the interface *Application*.

### 2.21.3 Methods and Properties

2.21.3.1 Property New

```
[propget, id(1)]
HRESULT New([in] BSTR Type, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Type | If *Type* equals *2d*, a 2 dimensional break point table shall be generated, if *Type* equals *3d*, a 3 dimensional break point table shall be generated, otherwise an error shall be generated. |
| *pVal | The index of the newly created Break Point Table in the collection. |

### 2.21.3.2 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | the Break Point Table with name *Name*. |
| *pVal | the index of the break point table with name *Name*, if it is found in the collection, and 0 otherwise. |

### 2.21.3.3 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

### 2.21.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.21.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.21.3.6 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the break point table data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which the data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.21.3.6.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.21.3.7 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.21.3.7.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

## 2.22 Interface "BreakPointTable2d"

### 2.22.1 Description

2.22.1.1 This interface will represent a two dimensional break point table.

### 2.22.2 Design

2.22.2.1 This interface shall be a dispatch interface.

2.22.2.2 This interface shall be an automation interface.

2.22.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *BreakPointTables*.

### 2.22.3 Methods and Properties

2.22.3.1 Property Name

```
// Name of the data record
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the break point table. When set, it must be non-empty and must only contain letters, digits, dot and underscores, otherwise an error shall be generated. |

2.22.3.2 Property Type

```
// Type of the break point table
[propget, id(303)]
HRESULT Type([out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the string "2d", thus distinguishing this interface from *BreakPointTable3d*. |

#### 2.22.3.3 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

#### 2.22.3.4 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

#### 2.22.3.5 Property ConfigLevel

```
// Configuration level of the Breakpoint table
[propget, id(3002)]
HRESULT ConfigLevel([out, retval] long *pVal);
[propput, id(3002)]
HRESULT ConfigLevel([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration level to which the data record belongs according to 2.1.3.4. The default value shall be the *ConfigLevel* value of the current configuration. If the property is set to a value which is not between 0 and this *ConfigLevel* value an error shall be generated. |

#### 2.22.3.6 Property Description

```
[propget, id(3005)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the description of the break point table. The default value shall be an empty string. |

#### 2.22.3.7 Property XEngineeringUnit

```
[propget, id(1)]
HRESULT XEngineeringUnit ([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT XEngineeringUnit ([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the engineering unit of the x values of the break point table. The default value shall be an empty string. When set, the engineering unit must be present in the engineering units' collection, otherwise an error shall be generated. |

#### 2.22.3.8 Property YEngineeringUnit

```
[propget, id(2)]
HRESULT YEngineeringUnit ([out, retval] BSTR *pVal);
[propput, id(2)]
HRESULT YEngineeringUnit ([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the engineering unit of the y values of the break point table. The default value shall be an empty string. When set, the engineering unit must be present in the engineering units' collection, otherwise an error shall be generated. |

### 2.22.3.9 Property Size

```
[propget, id(4)]
HRESULT Size([out, retval] long *pVal);
[propput, id(4)]
HRESULT Size([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the number of x and y values of the break point table. It shall be greater than or equal to 2. The default value shall be 2. |

### 2.22.3.10 Property XCoord

```
[propget, id(5)]
HRESULT XCoord([in] long Index, [out, retval] float *pVal);
[propput, id(5)]
HRESULT XCoord([in] long Index, [in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Index | If *Index* is smaller than 1 or greater than *Size*, an error shall be generated. |
| *pVal, newVal | the x value with position *Index* in the break point table. The default value shall be *Index*. |

### 2.22.3.11 Property YCoord

```
[propget, id(6)]
HRESULT YCoord([in] long Index, [out, retval] float *pVal);
[propput, id(6)]
HRESULT YCoord([in] long Index, [in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Index | If *Index* is smaller than 1 or greater than *Size*, an error shall be generated. |
| *pVal, newVal | the y value with position *Index* in the break point table. The default value shall be *Index*. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.22.3.12 Property Value

```
// Evaluation of the break point table.
[propget, id(8)]
HRESULT Value([in] float x, [out, retval] float *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| x | the x coordinate value to be evaluated in the Break Point Table. An error shall be generated if *x* is not between the minimum and the maximum x value. |
| *pVal | the value of the function defined by the break point table for the argument *x* using linear interpolation. |

### 2.22.3.13 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.22.4 Usage Conditions and Restrictions

2.22.4.1 The argument *Index* in the XCoord and YCoord properties will have to be between 1 and *Size*.

### 2.22.5 Example

2.22.5.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interfaces *BreakPointTables* and *BreakPointTable2d*.

```
' Establish the connection to the server.
Set app = CreateObject ("proDAS.Config")
Set bpts = app.BreakPointTables

Index = bpts.New ("2d")
Set bpt2 = bpts(Index)
bpt2.Size = 3
bpt2.XCoord (1) = 1.0
bpt2.YCoord (1) = 0.0
bpt2.XCoord (2) = 2.0
bpt2.YCoord (2) = 1.0
bpt2.XCoord (3) = 3.0
bpt2.YCoord (3) = 3.0

x = 1.5
MsgBox "Value at " & x & " is " & bpt2.Value (x)
```

## 2.23 Interface "BreakPointTable3d"

### 2.23.1 Description

2.23.1.1 This interface will represent a three dimensional break point table.

### 2.23.2 Design

2.23.2.1 This interface shall be a dispatch interface.

2.23.2.2 This interface shall be an automation interface.

2.23.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *BreakPointTables*.

### 2.23.3 Methods and Properties

2.23.3.1 Property Name

```
// Name of the data record
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the name of the break point table. When set, it must be non-empty and must only contain letters, digits, dot and underscores, otherwise an error shall be generated. |

2.23.3.2 Property Type

```
// Type of the break point table
[propget, id(303)]
HRESULT Type([out, retval] BSTR *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the string "3d", thus distinguishing this interface from *BreakPointTable2d*. |

### 2.23.3.3 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.23.3.4 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

### 2.23.3.5 Property ConfigLevel

```
// Configuration level of the Breakpoint table
[propget, id(3002)]
HRESULT ConfigLevel([out, retval] long *pVal);
[propput, id(3002)]
HRESULT ConfigLevel([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration level to which the data record belongs according to 2.1.3.4. The default value shall be the *ConfigLevel* value of the current configuration. If the property is set to a value which is not between 0 and this *ConfigLevel* value an error shall be generated. |

### 2.23.3.6 Property Description

```
[propget, id(3005)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the description of the break point table. The default value shall be an empty string. |

### 2.23.3.7 Property XEngineeringUnit

```
[propget, id(1)]
HRESULT XEngineeringUnit ([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT XEngineeringUnit ([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the engineering unit of the x values of the break point table. The default value shall be an empty string. When set, the engineering unit must be present in the engineering units' collection, otherwise an error shall be generated. |

#### 2.23.3.8 Property YEngineeringUnit

```
[propget, id(2)]
HRESULT YEngineeringUnit ([out, retval] BSTR *pVal);
[propput, id(2)]
HRESULT YEngineeringUnit ([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the engineering unit of the y values of the break point table. The default value shall be an empty string. When set, the engineering unit must be present in the engineering units' collection, otherwise an error shall be generated. |

#### 2.23.3.9 Property ZEngineeringUnit

```
[propget, id(3)]
HRESULT ZEngineeringUnit ([out, retval] BSTR *pVal);
[propput, id(3]
HRESULT ZEngineeringUnit ([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the engineering unit of the z values of the break point table. The default value shall be an empty string. When set, the engineering unit must be present in the engineering units' collection, otherwise an error shall be generated. |

#### 2.23.3.10 Property Size

```
[propget, id(4)]
HRESULT Size([out, retval] long *pVal);
[propput, id(4)]
HRESULT Size([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the number of x, y and z values of the break point table. It shall be greater than or equal to 4. The default value shall be 4. |

### 2.23.3.11 Property XCoord

```
[propget, id(5)]
HRESULT XCoord([in] long Index, [out, retval] float *pVal);
[propput, id(5)]
HRESULT XCoord([in] long Index,  [in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Index | If *Index* is smaller than 1 or greater than *Size*, an error shall be generated. |
| *pVal, newVal | the x value with position *Index* in the break point table. The default value shall be 0. |

### 2.23.3.12 Property YCoord

```
[propget, id(6)]
HRESULT YCoord([in] long Index, [out, retval] float *pVal);
[propput, id(6)]
HRESULT YCoord([in] long Index,  [in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Index | If *Index* is smaller than 1 or greater than *Size*, an error shall be generated. |
| *pVal, newVal | the y value with position *Index* in the break point table. The default value shall be 0. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.23.3.13 Property ZCoord

```
[propget, id(7)]
HRESULT ZCoord([in] long Index, [out, retval] float *pVal);
[propput, id(7)]
HRESULT ZCoord([in] long Index,  [in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Index | If *Index* is smaller than 1 or greater than *Size*, an error shall be generated. |
| *pVal, newVal | the z value with position *Index* in the break point table. The default value shall be 0. |

### 2.23.3.14 Property Value

```
// Evaluation of the break point table
[propget, id(8)]
HRESULT Value([in] float x, [in] float y,
      [out, retval] float *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| x | the x coordinate value to be evaluated in the Break Point Table. |
| y | the y coordinate value to be evaluated in the Break Point Table. |
| *pVal | the value of the function defined by the break point table for the arguments *x* and *y* using linear interpolation. An error shall be generated if interpolation is not possible. |

#### 2.23.3.15 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.23.4 Usage Conditions and Restrictions

2.23.4.1 The argument *Index* in the XCoord, YCoord and ZCoord properties will have to be between 1 and *Size*.

### 2.23.5 Example

2.23.5.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interfaces *BreakPointTables* and *BreakPointTable3d*.

```
' Establish the connection to the server.
Set app = CreateObject ("proDAS.Config")
Set bpts = app.BreakPointTables

Index = bpts.New ("3d")
Set bpt3 = bpts(Index)
Bpt3.Size = 18
Bpt3.XCoord (1) = 0.0
Bpt3.YCoord (1) = 0.0
Bpt3.ZCoord (1) = 1.0

Bpt3.XCoord (2) = 0.0
Bpt3.YCoord (2) = 1.0
Bpt3.ZCoord (2) = 2.0

Bpt3.XCoord (3) = 0.0
Bpt3.YCoord (3) = 3.0
Bpt3.ZCoord (3) = 3.0

Bpt3.XCoord (4) = 2.0
Bpt3.YCoord (4) = 0.0
Bpt3.ZCoord (4) = 1.5

Bpt3.XCoord (5) = 2.0
Bpt3.YCoord (5) = 1.0
Bpt3.ZCoord (5) = 2.5

Bpt3.XCoord (6) = 2.0
Bpt3.YCoord (6) = 3.0
Bpt3.ZCoord (6) = 3.5

x = 0
y = 1.5
MsgBox "Value at [" & x & ", " & y & "] is " & bpt3.Value (x,y)
```

## 2.24 Interface "TextOutputPages"

### 2.24.1 Description

2.24.1.1 This interface represents a collection of text output pages. A single text output page is defined in 2.25.

### 2.24.2 Design

2.24.2.1 This interface shall be a dispatch interface.

2.24.2.2 This interface shall be an automation interface.

2.24.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *TextOutputPage* interfaces in the collection.

2.24.2.4 This interface shall not be directly created. It shall be created by the *TextOutputPages* property of the interface *Application*.

### 2.24.3 Methods and Properties

2.24.3.1 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Title, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Title | the Text Output Page with title *Title*. |
| *pVal | the index of the Text Output Page with title *Title*, if it is found in the collection, and 0 otherwise. |

2.24.3.2 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

#### 2.24.3.3 Property New

```
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | The index of the newly created Text Output Page in the collection. |

#### 2.24.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

#### 2.24.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.24.3.6 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the Text Output Pages data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which the data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.24.3.6.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.24.3.7 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.24.3.7.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

## 2.25 Interface "TextOutputPage"

### 2.25.1 Description

2.25.1.1 This interface represents a text output page. A text output page is used to print text pages with inserted values or to display a dialog where values can be entered.

### 2.25.2 Design

2.25.2.1 This interface shall be a dispatch interface.

2.25.2.2 This interface shall be an automation interface.

2.25.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *TextOutputPages*.

### 2.25.3 Methods and Properties

2.25.3.1 Property Name

```
// Name of the data record
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the name of the text output page. The default value shall be an empty string. |

2.25.3.2 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.25.3.3 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.25.3.4 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

### 2.25.3.5 Property ConfigLevel

```
// Configuration level of the data record
[propget, id(3002)]
HRESULT ConfigLevel([out, retval] long *pVal);
[propput, id(3002)]
HRESULT ConfigLevel([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration level to which the data record belongs according to 2.1.3.4. The default value shall be the *ConfigLevel* value of the current configuration. If the property is set to a value which is not between 0 and this *ConfigLevel* value an error shall be generated. |

### 2.25.3.6 Property Description

```
[propget, id(3005)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the description of the text output page. The default value shall be an empty string. |

### 2.25.3.7 Property Rows

```
// Number of rows
[propget, id(3)]
HRESULT Rows([out, retval] long *pVal);
[propput, id(3)]
HRESULT Rows([in] long newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the number of rows of the page. If this number is less than 1 an error shall be generated. |

### 2.25.3.8 Property Cols

```
// Number of columns
[propget, id(4)]
HRESULT Cols([out, retval] long *pVal);
[propput, id(4)]
HRESULT Cols([in] long newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the number of columns of the page. If this number is less than 1 an error shall be generated. |

### 2.25.3.9 Property Entries

```
[propget, id(9)]
HRESULT Entries([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the collection of text output entries of the page. |

## 2.26 Interface "TextOutputEntries"

### 2.26.1 Description

2.26.1.1 This interface represents the collection of entries of a text output page.

### 2.26.2 Design

2.26.2.1 This interface shall be a dispatch interface.

2.26.2.2 This interface shall be an automation interface.

2.26.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *TextOutputEntry* interfaces in the collection.

2.26.2.4 This interface shall not be directly created. It shall be created by the *Entries* property of the interface *TextOutputPage*.

### 2.26.3 Methods and Properties

2.26.3.1 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

2.26.3.2 Property New

```
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | The index of the newly created Text Output Entry in the collection. |

### 2.26.3.3 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

## 2.27 Interface "TextOutputEntry"

### 2.27.1 Description

2.27.1.1 This interface represents an entry for a text output page. There are four different types of entry: fixed text, variable text, channel and form feed.

### 2.27.2 Design

2.27.2.1 This interface shall be a dispatch interface.

2.27.2.2 This interface shall be an automation interface.

2.27.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *TextOutputEntries*.

### 2.27.3 Methods and Properties

2.27.3.1 Property Type

```
// Type of the entry.
[propget, id(1)]
HRESULT Type([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT Type([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the type of the text output entry. Only the types *FixedText*, *VariableText*, *Channel* and *FormFeed* shall be allowed. If another type is set an error shall be generated. The default value shall be *FixedText*. |

2.27.3.2 Property Row

```
// Row of the entry.
[propget, id(2)]
HRESULT Row([out, retval] long *pVal);
[propput, id(2)]
HRESULT Row([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the row where the entry is placed. In the case of a form feed entry it shall be the row before the form feed. The default value shall be 1. |

### 2.27.3.3 Property Col

```
// Column of the entry.
[propget, id(3)]
HRESULT Col([out, retval] long *pVal);
[propput, id(3)]
HRESULT Col([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the column where the entry is placed. In the case of a form feed entry the value must be 1. The default value shall be 1. |

### 2.27.3.4 Property Text

```
// Text for fixed text entries.
[propget, id(4)]
HRESULT Text([out, retval] BSTR *pVal);
[propput, id(4)]
HRESULT Text([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the text for a *FixedText* entry only. The default shall be an empty string. |

#### 2.27.3.5 Property Field

```
// Field of TRSCDB for variable text entries.
[propget, id(5)]
HRESULT Field([out, retval] BSTR *pVal);
[propput, id(5)]
HRESULT Field([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the field of the test results database which is to be displayed. The following values shall be allowed: *TestName*, *TestDate*, *TestDescription*, *EngineType*, *EngineStandard*, *Customer*, *Build*, *SerialNumber*, *TestOperator1*, *TestOperator2*, *and TestEngineer*. The default shall be an empty string. This applies to a *VariableText* entry only. |

#### 2.27.3.6 Property Format

```
// Format for display.
[propget, id(6)]
HRESULT Format([out, retval] BSTR *pVal);
[propput, id(6)]
HRESULT Format([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the format used for display. It shall only be valid for *Channel* entries and for *VariableText* entries, otherwise an error shall be generated. The default value shall be *%f* for channel entries and *%s* for variable text entries. |

### 2.27.3.7 Property IsReadOnly

```
// Flags whether the entry is editable in a dialogue.
[propget, id(7)]
HRESULT IsReadOnly([out, retval] BOOL *pVal);
[propput, id(7)]
HRESULT IsReadOnly([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the entry can be changed in a dialogue. It shall only be valid for *Channel* entries and *VariableText* entries, otherwise an error shall be generated. The default value shall be *false*. |

### 2.27.3.8 Property EnterNew

```
// Flag for variable text entries with enumeration format
// whether a new value may be entered
[propget, id(8)]
HRESULT EnterNew([out, retval] BOOL *pVal);
[propput, id(8)]
HRESULT EnterNew([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether it is allowed to enter another value in a dialogue when an enumeration format is given for the entry. It shall only be valid for *Channel* entries, otherwise an error shall be generated. The default value shall be *false*. |

### 2.27.3.9 Property ChannelName

```
// Channel name for channel entries.
[propget, id(9)]
HRESULT ChannelName([out, retval] BSTR *pVal);
[propput, id(9)]
HRESULT ChannelName([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the channel associated with a *Channel* entry. It shall only be valid for *Channel* entries, otherwise an error shall be generated. The default shall be an empty string. |

### 2.27.3.10 Property LengthName

```
[propget, id(10)]
HRESULT LengthName([out, retval] long *pVal);
[propput, id(10)]
HRESULT LengthName([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | indicates the length in bytes of the channel name of the channel if it is to be displayed. This property shall only be valid for *Channel* entries, otherwise an error shall be generated. The default value for *LengthName* shall be *0*, indicating that the channel name is not to be displayed. |

### 2.27.3.11 Property LengthUnit

```
[propget, id(11)]
HRESULT LengthUnit([out, retval] long *pVal);
[propput, id(11)]
HRESULT LengthUnit([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | indicates the length in bytes of the corresponding unit of the channel if it is to be displayed. This property shall only be valid for *Channel* entries, otherwise an error shall be generated. The default value for *LengthUnit* shall be *0*, indicating that the channel unit is not to be displayed. |

### 2.27.3.12 Property LengthDescription

```
[propget, id(12)]
HRESULT LengthDescription([out, retval] long *pVal);
[propput, id(12)]
HRESULT LengthDescription([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | indicates the length in bytes of the channel description if it is to be displayed. This property shall only be valid for *Channel* entries, otherwise an error shall be generated. The default value for *LengthDescription* shall be *0*, indicating that the channel description is not to be displayed. |

### 2.27.3.13 Property LengthAlternateName

```
[propget, id(13)]
HRESULT LengthAlternateName([out, retval] long *pVal);
[propput, id(13)]
HRESULT LengthAlternateName([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | indicates the length in bytes of the alternate name of the channel if it is to be displayed. This property shall only be valid for *Channel* entries, otherwise an error shall be generated. The default value for *LengthAlternateName* shall be *0*, indicating that the channel alternate name is not to be displayed. |

### 2.27.3.14 Property DisplayValueInDialog

```
// Flags whether old values are to be displayed in a dialogue.
[propget, id(14)]
HRESULT DisplayValueInDialog([out, retval] BOOL *pVal);
[propput, id(14)]
HRESULT DisplayValueInDialog([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | Flags whether the current value is to be displayed in an editable field in a dialogue. This property shall only be valid for *Channel* entries, otherwise an error shall be generated. The default value shall be *true*. |

### 2.27.3.15 Property Min

```
// Allowed minimum value for input in dialogues.
[propget, id(15)]
HRESULT Min([out, retval] float *pVal);
[propput, id(15)]
HRESULT Min([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | denotes the minimum value allowed to enter in an editable field of a dialogue. It shall only be valid for *Channel* entries, otherwise an error shall be generated. The default value shall be -3.402823466e+38. |

### 2.27.3.16 Property Max

```
// Allowed maximum value for input in dialogues.
[propget, id(16)]
HRESULT Max([out, retval] float *pVal);
[propput, id(16)]
HRESULT Max([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | denotes the maximum value allowed to enter in an editable field of a dialogue. It shall only be valid for *Channel* entries, otherwise an error shall be generated. The default value shall be 3.402823466e+38. |

### 2.27.3.17 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | Flags whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

## 2.28 Interface "UserFunctions"

### 2.28.1 Description

2.28.1.1 This interface will represent a collection of user functions.

### 2.28.2 Design

2.28.2.1 This interface shall be a dispatch interface.

2.28.2.2 This interface shall be an automation interface.

2.28.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *UserFunction* interfaces in the collection.

2.28.2.4 This interface shall not be directly created. It shall be created by the *UserFunctions* property of the interface *Application*.

### 2.28.3 Methods and Properties

2.28.3.1 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | the User Function with name *Name*. |
| *pVal | the index of the User Function with name *Name*, if it is found in the collection, and 0 otherwise. |

2.28.3.2 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

#### 2.28.3.3 Property New

```
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | The index of the newly created User Function in the collection. |

#### 2.28.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

#### 2.28.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.28.3.6 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the User Functions data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which the data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.28.3.6.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.28.3.7 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.28.3.7.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.29 Interface "UserFunction"

### 2.29.1 Description

2.29.1.1 This interface will represent a single user function, consisting of a sequence of statements.

### 2.29.2 Design

2.29.2.1 This interface shall be a dispatch interface.

2.29.2.2 This interface shall be an automation interface.

2.29.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *UserFunctions*.

### 2.29.3 Methods and Properties

2.29.3.1 Property Name

```
// Name of the user function.
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the user function, uniquely identifying the user function within a configuration. The default value shall be an empty string. If it is set to an empty string or a *Name* already existing in the configuration defined by the *ConfigLevel* property an error shall be generated. |

2.29.3.2 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.29.3.3 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.29.3.4 Property IsEnabled

```
// Flags whether the user function is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the user function is to be used during a test. At the same time this will mean that the user function is validated when a consistency check is performed. The default value shall be *true*. |

### 2.29.3.5 Property ConfigLevel

```
// Configuration level of the user function
[propget, id(3002)]
HRESULT ConfigLevel([out, retval] long *pVal);
[propput, id(3002)]
HRESULT ConfigLevel([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration level to which the data record belongs according to 2.1.3.4. The default value shall be the *ConfigLevel* value of the current configuration. If the property is set to a value which is not between 0 and this *ConfigLevel* value an error shall be generated. |

### 2.29.3.6 Property Description

```
[propget, id(3005)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the description of the user function. The default value shall be an empty string. |

### 2.29.3.7 Property IsCyclic

```
// Flags whether the user function is to be executed
// cyclically.
[propget, id(4)]
HRESULT IsCyclic([out, retval] BOOL *pVal);
[propput, id(4)]
HRESULT IsCyclic([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the user function is to be executed cyclically, i.e. in every cycle of the User Function application. The default shall be *false*. |

#### 2.29.3.8 Property Statements

```
// Collection of statements.
[propget, id(9)]
HRESULT Statements( [out, retval] LPDISPATCH *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the collection of statements of the user function. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.30 Interface "UserFunctionStatements"

### 2.30.1 Description

2.30.1.1 This interface will represent the collection of statements of a single user function.

### 2.30.2 Design

2.30.2.1 This interface shall be a dispatch interface.

2.30.2.2 This interface shall be an automation interface.

2.30.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *UserFunctionStatement* interfaces in the collection.

2.30.2.4 This interface shall not be directly created. It shall be created by the *Statements* property of the interface *UserFunction*.

### 2.30.3 Methods and Properties

2.30.3.1 Method RemoveAll

```
// Delete all statements
[id(1)]
HRESULT RemoveAll();
```

2.30.3.2 Property New

```
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | The index of the newly created User Function Statement in the collection. |

### 2.30.3.3 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

## 2.31 Interface "UserFunctionStatement"

### 2.31.1 Description

2.31.1.1 This interface represents a statement of a user function (cf. 2.29).

### 2.31.2 Design

2.31.2.1 This interface shall be a dispatch interface.

2.31.2.2 This interface shall be an automation interface.

2.31.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *UserFunctionStatements*.

### 2.31.3 Methods and Properties

2.31.3.1 Property Type

```
// Type of the statement.
[propget, id(1)]
HRESULT Type([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT Type([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the type of the statement with the valid values *RealAssign*, *FunctionAssign*, *BooleanAssign*, *If*, *Else*, *ElseIf*, *EndIf*, *Goto*, *Label*, *Stop*, *Return*, *Comment* and empty string. The meaning of these values is defined in the Engineering Specification for User Functions. The default value shall be an empty string. |

2.31.3.2 Property AsComment

```
// Comment for the statement.
[propget, id(2)]
HRESULT AsComment([out, retval] BOOL *pVal);
[propput, id(2)]
HRESULT AsComment([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the statement is treated as a comment. The default value shall be *false*. |

### 2.31.3.3 Property Comment

```
// Comment for the statement.
[propget, id(3)]
HRESULT Comment([out, retval] BSTR *pVal);
[propput, id(3)]
HRESULT Comment([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the comment assigned to the statement. This property shall be valid for all types of statements. The default value shall be an empty string. |

### 2.31.3.4 Property Result

```
// Reference to result object.
[propget, id(4)]
HRESULT Result([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the result object of an assignment statement, representing the left-hand side of the statement. It shall only be valid for the statement types *RealAssign*, *FunctionAssign*, and *BooleanAssign*, otherwise an error shall be generated. |

### 2.31.3.5 Property Expression

```
// Reference to expression object.
[propget, id(5)]
HRESULT Expression([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the right-hand side of an assignment statement or the expression in an *If* or *ElseIf* statement. It shall only be valid for the statement types *RealAssign*, *FunctionAssign*, *BooleanAssign*, *If* and *ElseIf*, otherwise an error shall be generated. |

### 2.31.3.6 Property Label

```
// Label for goto statement.
[propget, id(6)]
HRESULT Label([out, retval] BSTR *pVal);
[propput, id(6)]
HRESULT Label([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the destination label for a *Goto* statement. If it is accessed for another statement type an error shall be generated. Note that it will not be checked whether the destination label exists. This will only be done by the User Functions module. The default value shall be an empty string. |

### 2.31.3.7 Property MaxJump

```
// Maximum number of jumps for goto statement.
[propget, id(7)]
HRESULT MaxJump([out, retval] long *pVal);
[propput, id(7)]
HRESULT MaxJump([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the maximum number of times that a *Goto* statement may be executed. If it is accessed for another statement type an error shall be generated. The default shall be 1. If this property is set to a non-positive value an error shall be generated. This property will be used to limit the number of executions of a loop programmed with a *Goto* statement. |

2.31.3.8 Property LabelName

```
// Name of a label statement
[propget, id(8)]
HRESULT LabelName([out, retval] BSTR *pVal);
[propput, id(8)]
HRESULT LabelName([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of a label. If it is accessed for a statement type other than *Label* an error shall be generated. The default shall be an empty string. |

2.31.3.9 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.31.4 Usage Conditions and Restrictions

2.31.4.1 The *Result* property may only be accessed for *RealAssign*, *FunctionAssign*, and *BooleanAssign* statements.

2.31.4.2 The *Expression* property may only be accessed for *RealAssign*, *FunctionAssign*, *BooleanAssign*, *If* and *ElseIf* statements.

2.31.4.3 The *Label* and the *MaxJump* properties may only be accessed for *Goto* statements.

2.31.4.4 The *LabelName* property may only be accessed for *Label* statements.

## 2.32 Interface "UserFunctionResult"

### 2.32.1 Description

2.32.1.1 This interface represents the left-hand side of an assignment statement (cf. 2.31) of a user function.

### 2.32.2 Design

2.32.2.1 This interface shall be a dispatch interface.

2.32.2.2 This interface shall be an automation interface.

2.32.2.3 This interface shall not be directly created. It shall be created by the *Result* property of the interface *UserFunctionStatement*.

### 2.32.3 Methods and Properties

2.32.3.1 Property Name

```
// Name of the result variable.
[propget, id(1)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the variable used as the left-hand side of an assignment statement. It will have to be a channel name for global scope, but this shall not be checked. It shall be possible to have a variable of local scope with the same name as a channel. |

### 2.32.3.2 Property IsScopeLocal

```
// Flag if the variable has local scope.
[propget, id(2)]
HRESULT IsScopeLocal ([out, retval] BOOL *pVal);
[propput, id(2)]
HRESULT IsScopeLocal ([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the variable has local scope, i.e. it is not a channel and exists only within the user function. The default value shall be *false*. |

### 2.32.3.3 Property IsVolatile

```
// Flag if the variable is volatile.
[propget, id(3)]
HRESULT IsVolatile ([out, retval] BOOL *pVal);
[propput, id(3)]
HRESULT IsVolatile ([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the local variable will be (re-) initialised before each execution of the user function (i.e. the variable is volatile); or whether it will keep its value between successive cycles (i.e. non-volatile). Note: this property would not be applied to channel variables. The default value shall be *true*. |

### 2.32.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

## 2.33 Interface "UserFunctionExpression"

### 2.33.1 Description

2.33.1.1 This interface represents an expression of a statement (cf. 2.31) in a user function. This expression is either the right hand side of an assignment statement or a Boolean expression of an *if* or *else if* statement.

### 2.33.2 Design

2.33.2.1 This interface shall be a dispatch interface.

2.33.2.2 This interface shall be an automation interface.

2.33.2.3 This interface shall not be directly created. It shall be created by the *Expression* property of the interface *UserFunctionStatement*.

### 2.33.3 Methods and Properties

2.33.3.1 Property Operator

```
// Name of the operator.
[propget, id(1)]
HRESULT Operator([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT Operator([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the string operator type of the expression with one of the following values. |

2.33.3.1.1 The ***Operator*** values are:

- empty string (identity),
- *Initialisation*, *AllEvents* (Boolean without arguments)
- *NOT* (unary Boolean),
- *OR*, *AND*, *XOR*, *NOR*, *NAND*, *EQUIV*, *NEQUIV* (binary Boolean),
- *IsNaN* (unary real with Boolean result),
- <, >, <=, >=, ==, != (binary real with Boolean result),
- *NaN, Rand, Time, Date (real without argument)*
- *ChangeSign*,
- +, -, *, /, ^, *atan2, fmod, hypot, max, min, IntervalMin, IntervalMax, BitTest, TimeSince(date,time)* (binary real),
- *TimeDiff(date1,time1,date2,time2)* (time difference),
- *Lookup*, *LookupInv*, *Lookup3d*, *Lookup3dInvX*, *Lookup3dInvY* (break point table interpolation functions),
- *GRAIN*, *Sr2*, *Sr5* (aerodynamic functions).

2.33.3.1.2 The meaning of these operators is explained in the Engineering Specification for User Functions. The default shall be an empty string.

### 2.33.3.2 Property Operands

```
// Collection of operands.
[propget, id(2)]
HRESULT Operands([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the collection of operands, the size of which will depend on the operator. |

### 2.33.3.3 Property Function

```
// Function to be applied to the result of an operator
// with Boolean result
[propget, id(3)]
HRESULT Function([out, retval] BSTR *pVal);
[propput, id(3)]
HRESULT Function([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the function to be applied to the result of an operator with Boolean result. The default value shall be an empty string, indicating that no function is to be applied. The property shall have one of the values *FirstTime* or *LastTime*. If the property is set to another value or if the property is accessed for an operator with a float result an error shall be generated. |

#### 2.33.3.4 Property BreakPointTable

```
// Table for interpolation function
[propget, id(4)]
HRESULT BreakPointTable([out, retval] BSTR *pVal);
[propput, id(4)]
HRESULT BreakPointTable([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the name of a 2d or 3d break point table used for an Interpolation function. The Interpolation function type shall define whether a 2d or a 3d table is required. It shall be mandatory for this type i.e. Interpolation function and not permitted for others. An error shall be generated if it is accessed for other operators. |

### 2.33.4 Usage Conditions and Restrictions

2.33.4.1 The *BreakPointTable* property may only be accessed for *Interpolation* operators.

2.33.4.2 The *Operator* property may only be set to one of the values defined in 2.33.3.1.1.

2.33.4.3 The *Function* property may only accessed for operators with Boolean results. It may only have the values defined in 2.33.3.3.

## 2.34 Interface "UserFunctionOperands"

### 2.34.1 Description

2.34.1.1 This interface represents the collection of operands of an expression of a statement (cf. 2.33) in a user function.

### 2.34.2 Design

2.34.2.1 This interface shall be a dispatch interface.

2.34.2.2 This interface shall be an automation interface.

2.34.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *UserFunctionOperand* interfaces in the collection.

2.34.2.4 This interface shall not be directly created. It shall be created by the *Operands* property of the interface *UserFunctionExpression*.

### 2.34.3 Methods and Properties

2.34.3.1 Property New

```
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | The index of the newly created User Function Operand in the collection. |

2.34.3.2 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

## 2.35 Interface "UserFunctionOperand"

### 2.35.1 Description

2.35.1.1 This interface represents an operand in an expression (cf.2.33), which belongs to the statement in a user function.

### 2.35.2 Design

2.35.2.1 This interface shall be a dispatch interface.

2.35.2.2 This interface shall be an automation interface.

2.35.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *UserFunctionOperands*.

### 2.35.3 Methods and Properties

2.35.3.1 Property Name

```
// Name of the operand, if not a literal.
[propget, id(1)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the channel or the local variable which serves as the operand. The default value shall be an empty string, indicating that the operand is a literal and the *Value* or *ValueBool* property is valid or one of the functions *Initialisation* or *AllEvents* is set. |

### 2.35.3.2 Property Value

```
// Value of the operator, if float.
[propget, id(2)]
HRESULT Value([out, retval] float *pVal);
[propput, id(2)]
HRESULT Value([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the float value, if the operand is a float literal, i.e. if the *Name* property is an empty string and the operand comes from an expression with an operator with float operands. The default value shall be 0.0. |

### 2.35.3.3 Property ValueBool

```
// Value of the operator, if Boolean.
[propget, id(3)]
HRESULT ValueBool([out, retval] BOOL *pVal);
[propput, id(3)]
HRESULT ValueBool([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the Boolean value, if the operand is a Boolean literal, i.e. if the *Name* property is an empty string and the operand comes from an expression with an operator with Boolean operands. The default value shall be *false*. This property is irrelevant for the functions *Initialisation* and *AllEvents*. |

#### 2.35.3.4 Property IsScopeLocal

```
// Flag if the variable has local scope.
[propget, id(4)]
HRESULT IsScopeLocal ([out, retval] BOOL *pVal);
[propput, id(4)]
HRESULT IsScopeLocal ([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the variable has local scope, i.e. it is not a channel and exists only within the user function. The default value shall be *false*. |

#### 2.35.3.5 Property Function

```
[propget, id(5)]
HRESULT Function([out, retval] BSTR *pVal);
[propput, id(5)]
HRESULT Function([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the function to be applied to the literal, channel value or local variable. It shall have one of the values *abs, acos, asin, atan, ceil, cos, cosh, exp, fabs, floor, log, log10, round, sin, sinh, sqrt, square, tan, tanh* or an empty string. The meaning of these functions is defined in the Engineering Specification for User Functions. The default shall be an empty string, denoting the identity function. An error shall be generated if the property is set to an incorrect value. |

### 2.35.4 Usage Conditions and Restrictions

2.35.4.1 The *Function* property may only be set in the case of float operands.

2.35.4.2 The property *Value* may only be accessed if the operand has been accessed from an expression with float operands.

2.35.4.3 The property *ValueBool* may only be accessed if the operand has been accessed from an expression with Boolean operands.

2.35.4.4 The property *IsScopeLocal* may only be accessed if the *Name* property is not set to an empty string.

## 2.36 Interface "Macros"

### 2.36.1 Description

2.36.1.1 This interface represents a collection of macros. A single macro is defined in 2.37.

### 2.36.2 Design

2.36.2.1 This interface shall be a dispatch interface.

2.36.2.2 This interface shall be an automation interface.

2.36.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *Macro* interfaces in the collection.

2.36.2.4 This interface shall not be directly created. It shall be created by the *Macros* property of the interface *Application*.

### 2.36.3 Methods and Properties

2.36.3.1 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | the Macro with name *Name*. |
| *pVal | the index of the Macro with name *Name*, if it is found in the collection, and 0 otherwise. |

2.36.3.2 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

### 2.36.3.3 Property New

```
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | The index of the newly created Macro in the collection. |

### 2.36.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.36.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.36.3.6 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the Macro data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which the data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.36.3.6.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.36.3.7 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.36.3.7.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

## 2.37 Interface "Macro"

### 2.37.1 Description

2.37.1.1 This interface represents a VBS macro used for the automation of tasks. It does not contain the macro itself but only points to a file containing it.

### 2.37.2 Design

2.37.2.1 This interface shall be a dispatch interface.

2.37.2.2 This interface shall be an automation interface.

2.37.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *Macros*.

### 2.37.3 Methods and Properties

2.37.3.1 Property Name

```
// Name of the macro
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the macro as it will be displayed in the GUI. The default value shall be an empty string. |

2.37.3.2 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.37.3.3 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.37.3.4 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

### 2.37.3.5 Property ConfigLevel

```
// Configuration level of the data record
[propget, id(3002)]
HRESULT ConfigLevel([out, retval] long *pVal);
[propput, id(3002)]
HRESULT ConfigLevel([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration level to which the data record belongs according to 2.1.3.4. The default value shall be the *ConfigLevel* value of the current configuration. If the property is set to a value which is not between 0 and this *ConfigLevel* value an error shall be generated. |

### 2.37.3.6 Property FileName

```
// Name of the macro file
[propget, id(2)]
HRESULT FileName([out, retval] BSTR *pVal);
[propput, id(2)]
HRESULT FileName([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the file where the VBS macro is stored. This file name shall have the extension *.vbs*, otherwise an error shall be generated when the property is set. When the property is set and the file does not exist, an error shall be generated. The default value shall be an empty string. |

2.37.3.6.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

#### 2.37.3.7 Property Description

```
// Description of the macro
[propget, id(3005)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the description of the macro. The default value shall be an empty string. |

#### 2.37.3.8 Property Categories

```
// Category of the macro
[propget, id(7)]
HRESULT Categories([out, retval] LPDISPATCH *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the list of categories the macro belongs to using the interface *Strings*. |

### 2.37.4 Example

2.37.4.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interfaces *Macros* and *Macro*.

```
' Establish the connection to the server.
Set App = CreateObject ("proDAS.Config")

' Access the macro collection
Set Macros = App.Macros

' Show all macros.
for i = 1 to Macros.Count
   Set Macro = Macros (i)
   MsgBox "Macro " & i & vbLf &_
          "Name=" & Macro.Name & vbLf &_
          "ConfigLevel=" & Macro.ConfigLevel & vbLf &_
          "Description=" & Macro.Description & vbLf &_
          "File=" & Macro.FileName
next
```

## 2.38 Interface "TransientLogDefs"

### 2.38.1 Description

2.38.1.1 This interface represents a collection of transient log definitions. A single transient log definition is defined in 2.39.

### 2.38.2 Design

2.38.2.1 This interface shall be a dispatch interface.

2.38.2.2 This interface shall be an automation interface.

2.38.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *TransientLogDef* interfaces in the collection.

2.38.2.4 This interface shall not be directly created. It shall be created by the *TransientLogDefs* property of the interface *Application*.

### 2.38.3 Methods and Properties

2.38.3.1 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | the Transient Log Definition with name *Name*. |
| *pVal | the index of the Transient Log Definition with name *Name*, if it is found in the collection, and 0 otherwise. |

2.38.3.2 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

#### 2.38.3.3 Property New

```
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | The index of the newly created Transient Log Definition in the collection. |

#### 2.38.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

#### 2.38.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.38.3.6 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the Transient Log Definition data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which the data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.38.3.6.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.38.3.7 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.38.3.7.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

## 2.39 Interface "TransientLogDef"

### 2.39.1 Description

2.39.1.1 This interface represents a transient log definition, which can be used by the RTE to log the values of some channels.

### 2.39.2 Design

2.39.2.1 This interface shall be a dispatch interface.

2.39.2.2 This interface shall be an automation interface.

2.39.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *TransientLogDefs*.

### 2.39.3 Methods and Properties

2.39.3.1 Property Name

```
// Name of the transient log definition
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the name of the transient log definition as it will be displayed in the GUI. The default value shall be an empty string. |

2.39.3.2 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.39.3.3 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

### 2.39.3.4 Property ConfigLevel

```
// Config level of the transient log definition
[propget, id(3002)]
HRESULT ConfigLevel([out, retval] long *pVal);
[propput, id(3002)]
HRESULT ConfigLevel([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the configuration level to which the data record belongs according to 2.1.3.4. The default value shall be the *ConfigLevel* value of the current configuration. If the property is set to a value which is not between 0 and this *ConfigLevel* value an error shall be generated. |

### 2.39.3.5 Property FormatVersion

```
[propget, id(8)]
HRESULT FormatVersion([out, retval] BSTR *pVal);
[propput, id(8)]
HRESULT FormatVersion([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the version number of the format for the transient log. The default value shall be “2.0”. |

### 2.39.3.6 Property Description

```
[propget, id(3005)]
HRESULT Description([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description([in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the description of the transient log definition. The default value shall be an empty string. |

### 2.39.3.7 Property Duration

```
[propget, id(3)]
HRESULT Duration([out, retval] long *pVal);
[propput, id(3)]
HRESULT Duration([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the duration of the log in milliseconds. The default value shall be 0. If 0 is set an error shall be generated. |

### 2.39.3.8 Property Rate

```
[propget, id(4)]
HRESULT Rate([out, retval] float *pVal);
[propput, id(4)]
HRESULT Rate([in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the logging rate in Hz. The default value shall be 1.0. |

### 2.39.3.9 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

### 2.39.3.10 Method ImportTabDelimited

```
// Import tab delimited files
[id(10)]
HRESULT ImportTabDelimited ([in] BSTR FileName);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | Imports the data contained in the file named by *FileName* in tab delimited format (cf. Functional Requirements Document for proDAS ) into the transient log definition. This data shall define a single Transient Log file. If the file does not exist, the file cannot be opened or the syntax is incorrect an error shall be generated. The imported data will not be checked because this will happen when the data are written back. |

2.39.3.10.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

2.39.3.10.2 If the Transient Log Definition already exists, it will be replaced by the imported one.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.39.3.11 Method ExportTabDelimited

```
// Export tab delimited files
[id(11)]
HRESULT ExportTabDelimited ([in] BSTR FileName);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the data of the transient log definition into the file named by *FileName* in tab delimited format (cf. Functional Requirements Document for proDAS ). This data shall define a single Transient Log file. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |

2.39.3.11.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.39.3.12 Method ImportTabDelimitedText

```
// Import data from a string in tab delimited format.
[id(13)]
HRESULT ImportTabDelimitedText([in] VARIANT Text);
```

| **Argument Name** | **Description** |
|---|---|
| Text | Byte array containing the transient log definition in tab delimited format. |

### 2.39.3.13 Method ExportTabDelimitedText

```
// Export data to a string in tab delimited format.
[id(14)]
HRESULT ExportTabDelimitedText([out] VARIANT* pText);
```

| **Argument Name** | **Description** |
|---|---|
| *pText | Byte array containing the transient log definition in tab delimited format. |

### 2.39.3.14 Property Channels

```
// Channels to be logged
[propget, id(12)]
HRESULT Channels([out, retval] LPDISPATCH *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the list of channel names belonging to the transient log definition. The return value is a *Strings* interface. |

### 2.39.4 Example

2.39.4.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interface *TransientLogDefs* and *TransientLogDef*.

```
' Establish the connection to the server.
Set App = CreateObject ("proDAS.Config")

' Access the collection of transient log definitions.
Set Logs = App.TransientLogDefs

' Save the number of transient log definitions.
count = Logs.Count

' Create a new transient log definition
Index = Logs.New
if Logs.Count <> count + 1 then
   MsgBox "Transient log definition not correctly created."
end if
Set LogDef = Logs (Index)
LogDef.Name = "MyLog"
LogDef.Description = "This is a test log."
LogDef.ConfigLevel = 2     ' engine type specific
LogDef.Duration = 10000    ' in ms
LogDef.Rate = 50           ' in Hz

' Find the newly created log.
Index = Logs.Find("MyLog")
if Index = 0 then
   MsgBox "Newly created log not found."
else
   set LogDef = Logs (Index)

   ' Show the new log definition.
   MsgBox "New log: Name=" & LogDef.Name & vbLf & _
          "Description=" & LogDef.Description & vbLf & _
          "ConfigLevel=" & LogDef.ConfigLevel & vbLf & _
          "Duration=" & LogDef.Duration & vbLf & _
          "Rate=" & LogDef.Rate & vbLf

   'Delete the new transient log definition again.
   Logs.Delete (Index)
end if
```

## 2.40 Interface "Options"

### 2.40.1 Description

2.40.1.1 This interface handles options of different data types. These options are configuration dependent. They may also be installation dependent. Options are used for data exchange between applications and to store miscellaneous data persistently.

### 2.40.2 Design

2.40.2.1 This interface shall be a dispatch interface.

2.40.2.2 This interface shall be an automation interface.

2.40.2.3 Unlike the other configuration data, options shall be available for other clients immediately. This will enable data exchange between applications.

2.40.2.4 This interface shall implement the collection interface for *BSTR*, returning the option names in the collection.

2.40.2.5 This interface shall not be directly created. It shall be created by the *Options* property of the interface *Application*.

2.40.2.6 There shall be a possibility to save options explicitly on a client’s request. In addition, options shall be written by the ConfigServer automatically, before the server is shutting-down.

### 2.40.3 Methods and Properties

2.40.3.1 Property Count

```
// Number of Strings
[propget, id(1)]
HRESULT Count([out, retval] long *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal | number of strings in the collection. |

### 2.40.3.2 Property _NewEnum

```
// Accessing the enumeration interface
[propget, id(DISPID_NEWENUM)]
HRESULT _NewEnum([out, retval] LPUNKNOWN *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | an interface *IEnumVARIANT*, used for enumerating the members of the collection. |

### 2.40.3.3 Property Item

```
// Accessing an item by index
[propget, id(DISPID_VALUE)]
HRESULT Item([in] long Index, [out, retval] BSTR *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Index | Index, where indexing shall start with 1. If *Index* is not between 1 and *Count* an error *E_INVALIDARG* shall be generated. |
| *pVal | the name of the item with index *Index*. |

### 2.40.3.4 Property Integer

```
[propget, id(2)]
HRESULT Integer([in] BSTR Name, [out, retval] long *pVal);
[propput, id(2)]
HRESULT Integer([in] BSTR Name, [in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | The name *Name* of the Option. The name shall be unique for a given type, i.e. there may be options with the same name, but of different types within the same configuration. |
| *pVal, newVal | The value of the integer option with name *Name*. |

#### 2.40.3.5 Property String

```
[propget, id(3)]
HRESULT String([in] BSTR Name, [out, retval] BSTR *pVal);
[propput, id(3)]
HRESULT String([in] BSTR Name, [in] BSTR newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | The name *Name* of the Option. The name shall be unique for a given type, i.e. there may be options with the same name, but of different types within the same configuration. |
| *pVal, newVal | The value of the string option with name *Name*. |

#### 2.40.3.6 Property Float

```
[propget, id(4)]
HRESULT Float([in] BSTR Name, [out, retval] float *pVal);
[propput, id(4)]
HRESULT Float([in] BSTR Name, [in] float newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | The name *Name* of the Option. The name shall be unique for a given type, i.e. there may be options with the same name, but of different types within the same configuration. |
| *pVal, newVal | The value of the float option with name *Name*. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.40.3.7 Property Boolean

```
[propget, id(5)]
HRESULT Boolean([in] BSTR Name, [out, retval] BOOL *pVal);
[propput, id(5)]
HRESULT Boolean([in] BSTR Name, [in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | The name *Name* of the Option. The name shall be unique for a given type, i.e. there may be options with the same name, but of different types within the same configuration. |
| *pVal, newVal | The value of the Boolean option with name *Name*. |

### 2.40.3.8 Property IsLocalWrite (Obsolete)

```
// Flag indicating whether the options written only apply
// to the local installation
[propget, id(6)]
HRESULT IsLocalWrite([out, retval] BOOL *pVal);
[propput, id(6)]
HRESULT IsLocalWrite([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | (Obsolete) whether the operations of the properties *Integer*, *String*, *Float* and *Boolean* only apply to the current installation i.e. the Configuration Server computer when *IsLocalWrite* is *true* or to all installations i.e. all computers when *IsLocalWrite* is *false*. The default value shall be *false*. When reading an *Integer*, *String*, *Float* or *Boolean* property, the local property shall take precedence i.e. the value written when *IsLocalWrite* was *true* for the current installation. |

### 2.40.3.9 Property IntegerDefault

```
// Default values for the options of the different types
// if no values are stored.
[propget, id(7)]
HRESULT IntegerDefault ([out, retval] long *pVal);
[propput, id(7)]
HRESULT IntegerDefault ([in] long newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the default value for integer options. |

### 2.40.3.10 Property StringDefault

```
// Default values for the options of the different types
// if no values are stored.
[propget, id(8)]
HRESULT StringDefault ([out, retval] BSTR *pVal);
[propput, id(8)]
HRESULT StringDefault ([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the default value for string options. |

### 2.40.3.11 Property FloatDefault

```
// Default values for the options of the different types
// if no values are stored.
[propget, id(9)]
HRESULT FloatDefault ([out, retval] float *pVal);
[propput, id(9)]
HRESULT FloatDefault ([in] float newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the default value for float options. |

### 2.40.3.12 Property BooleanDefault

```
// Default values for the options of the different types
// if no values are stored.
[propget, id(10)]
HRESULT BooleanDefault ([out, retval] BOOL *pVal);
[propput, id(10)]
HRESULT BooleanDefault ([in] BOOL newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | the default value for Boolean options. |

### 2.40.3.13 Property ConfigLevelWrite

```
// Scope for write operations.
[propget, id(14)]
HRESULT ConfigLevelWrite([out, retval] long *pVal);
[propput, id(14)]
HRESULT ConfigLevelWrite([in] long newVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal, newVal | a configuration level according to 2.1.3.4. This configuration level shall then apply to all put operations for the properties *Integer*, *String*, *Float* and *Boolean*. It shall have no effect on get operations. |

### 2.40.3.14 Method Remove

```
// Delete an option.
[id(15)]
HRESULT Remove([in] BSTR Name);
```

| **Argument Name** | **Description** |
|---|---|
| Name | the option with name *Name* from the configuration defined by *ConfigLevelWrite* and according to the flag *IsLocalWrite*. No error shall be generated if the option does not exist. |

2.40.3.14.1 All Options with the same name regardless of their type will be removed.

### 2.40.3.15 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel,
      [in, optional, defaultvalue("FALSE")] BOOL Replace);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall denote the configuration level to which the data records are imported, the default -1 denoting the configuration level of the current configuration. The *ConfigLevel* must be less than or equal to the *ConfigLevel* of the current configuration. |
| Replace | The *Replace* flag, with default *false*, shall indicate whether the existing data are replaced or the new data are added. |

2.40.3.15.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.40.3.16 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName,
      [in, optional, defaultvalue(-1)] long ConfigLevel);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the Option data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |
| ConfigLevel | The parameter *ConfigLevel* shall indicate the configuration level from which the data records are to be exported, the default value -1 indicating that all the levels are to be exported. |

2.40.3.16.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

2.40.3.17 Method SaveAll

```
// Write all options to the database
[id(18)]
HRESULT SaveAll(void);
```

### 2.40.4 Usage Conditions and Restrictions

2.40.4.1 If an option already exists when the property is written, its value shall be overwritten. If the option does not exist yet when a property is written the property shall be created within the configuration defined by the *ConfigLevelWrite* property.

2.40.4.2 If an option is read and does not exist yet, the option shall be created within the current configuration with the default value of the appropriate type and the default value shall be returned.

2.40.4.3 *ConfigLevelWrite* will have to be called before it takes effect in a put operation.

2.40.4.4 The default properties will have to be set before they take effect in a get operation.

2.40.4.5 The Option *Name* will be defined by a client application. The following defines the naming convention to be used. This convention will help to maintain a good discipline on the ever growing list of options.

2.40.4.5.1 The Option *Name* will be preceded with the initials of its module in uppercase followed by the option name in an uppercase and lowercase combination. (i.e.: <MODULEINITIAL><OptionName>).

2.40.4.5.2 The module initials <MODULEINITIAL> will consist of 3 to 4 letters in uppercase (e.g. Channel Editor: CHE, Hardware Editor: HAE, etc.).

2.40.4.5.3 The option name <OptionName> will consist of a descriptive name composed of several words. Each word will begin with an uppercase letter and will be followed by lowercase letters. (e.g. The Font Face Name definition for the Channel Editor will be defined as CHEFontFaceName and could be set to “Times New Roman” and the Font Face Name definition for the Hardware Editor will be defined as HAEFontFaceName and could be set to “Arial”).

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.40.5 Persistent Data

2.40.5.1 The properties *Integer*, *String*, *Float* and *Boolean* shall be stored persistently in the correct configuration and if applicable, together with the information to which installation they belong.

### 2.40.6 Example

2.40.6.1 The following is an example of a VBS client which illustrates how to use the properties and methods of the interface Options.

```
' Establish the connection to the server.
Set app = CreateObject ("proDAS.Config")

' Access the collection of options.
Set Options = app.Options

' Delete an option.
name = "MyOption"
options.Remove name

' Set the default integer value.
options.IntegerDefault = 77
MsgBox name & "=" & options.Integer (name)

' Set an integer option.
options.Integer (name) = 88

' Set the string option with the same name.
options.String (name) = "MyValue"

MsgBox name & "=" & options.Integer (name) & vbLf & _
       name & "=" & options.String (name)
```

## 2.41 Interface "SiteSpecificChannelAttributes"

### 2.41.1 Description

2.41.1.1 This interface represents a collection of site specific channel attributes. A single site specific channel attribute is defined in 2.42.

### 2.41.2 Design

2.41.2.1 This interface shall be a dispatch interface.

2.41.2.2 This interface shall be an automation interface.

2.41.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *SiteSpecificChannelAttribute* interfaces in the collection.

2.41.2.4 This interface shall not be directly created. It shall be created by the *SiteSpecificChannelAttributes* property of the interface *Application*.

### 2.41.3 Methods and Properties

2.41.3.1 Property Find

```
// Find a data record.
[propget, id(4005)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| Name | the site specific channel attribute with name *Name*. |
| *pVal | the index of the data record with name *Name*, if it is found in the collection, and 0 otherwise. |

2.41.3.2 Property IsConsistent

```
// Check of consistency.
[propget, id(1002)]
HRESULT IsConsistent([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | *true* means that the data are consistent, *false* that they are not. In the latter case it will be possible to read any errors via the *Errors* interface. |

### 2.41.3.3 Property New

```
// Create a data record.
[propget, id(1003)]
HRESULT New([out, retval] long *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | The index of the newly created site specific channel attribute in the collection. |

### 2.41.3.4 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(1005)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.41.3.5 Property LastModificationDate

```
// Last modification date
[propget, id(4000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| **Argument Name** | **Description** |
|---|---|
| *pVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.41.3.6 Method Export

```
// Export the data into a file
[id(4001)]
HRESULT Export([in] BSTR FileName);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | exports the SiteSpecificChannelAttributes data into the file named by *FileName* in XML format. If the file already exists it shall be overwritten. If the file cannot be created an error shall be generated. |

2.41.3.6.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.41.3.7 Method Import

```
// import the data contained in the file
[id(4002)]
HRESULT Import([in] BSTR FileName,
      [in, optional, defaultvalue("Append")] BSTR Policy);
```

| **Argument Name** | **Description** |
|---|---|
| FileName | A file name accessible by the Configuration Server. If the file does not exist, the file cannot be opened, or the XML syntax is incorrect, an error shall be generated. |
| Policy | The import policy *Policy*, shall indicate whether the existing data are replaced, merged or appended. |

2.41.3.7.1 Note that the file name FileName shall include the full path and be accessible by the Configuration Server, i.e. the file name shall be reachable over the network from the Client and the Server computer.

### 2.42 Interface "SiteSpecificChannelAttribute"

#### 2.42.1 Description

2.42.1.1 This interface represents a site specific channel attribute.

#### 2.42.2 Design

2.42.2.1 This interface shall be a dispatch interface.

2.42.2.2 This interface shall be an automation interface.

2.42.2.3 This interface shall not be directly created. It shall be created by the *Item* property of the interface *SiteSpecificChannelAttributes*.

#### 2.42.3 Methods and Properties

2.42.3.1 Property Name

```
// Name of the data record
[propget, id(3004)]
HRESULT Name([out, retval] BSTR *pVal);
[propput, id(3004)]
HRESULT Name([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the name of the site specific channel attribute. The default value shall be an empty string. An error shall be generated if the name is set to an empty string. |

2.42.3.2 Property IsEnabled

```
// Flags whether the data record is enabled
[propget, id(3001)]
HRESULT IsEnabled([out, retval] BOOL *pVal);
[propput, id(3001)]
HRESULT IsEnabled([in] BOOL newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | whether the data record is to be used during a test. At the same time this will mean that the data record is validated when a consistency check is performed. The default value shall be *true*. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.42.3.3 Property IsChanged

```
// Flags whether the data have been changed
[propget, id(2000)]
HRESULT IsChanged([out, retval] BOOL *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | whether the data have been changed. It shall initially be set to *false*. A change may concern data which are directly accessible by the interface or indirectly by an interface obtained from it. |

### 2.42.3.4 Property DisplayName

```
[propget, id(1)]
HRESULT DisplayName ([out, retval] BSTR *pVal);
[propput, id(1)]
HRESULT DisplayName ([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | The DisplayName of the SiteSpecificChannelAttribute. This name can be used as a header when displaying the site specific attributes of a channel. |

### 2.42.3.5 Property DataType

```
[propget, id(2)]
HRESULT DataType ([out, retval] BSTR *pVal);
[propput, id(2)]
HRESULT DataType ([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | The DataType of the SiteSpecificChannelAttribute. This can be one of the following: String, Enumeration, Boolean or Float. |

### 2.42.3.6 Property DefaultValue

```
[propget, id(3)]
HRESULT DefaultValue ([out, retval] BSTR *pVal);
[propput, id(3)]
HRESULT DefaultValue ([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | The DefaultValue of the SiteSpecificChannelAttribute. |

#### 2.42.3.7 Property DisplayFormat

```
[propget, id(4)]
HRESULT DisplayFormat ([out, retval] BSTR *pVal);
[propput, id(4)]
HRESULT DisplayFormat ([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | The DisplayFormat of the SiteSpecificChannelAttribute. Mandatory for enumeration type only, otherwise the default is %s or %f |

#### 2.42.3.8 Property Description

```
[propget, id(3005)]
HRESULT Description ([out, retval] BSTR *pVal);
[propput, id(3005)]
HRESULT Description ([in] BSTR newVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | The Description of the SiteSpecificChannelAttribute. Optional. |

#### 2.42.3.9 Property LastModificationDate

```
// Last modification date
[propget, id(3000)]
HRESULT LastModificationDate([out, retval] DATE *pVal);
```

| Argument Name | Description |
|---|---|
| *pVal, newVal | the time when the data administrated by the interface have last been changed. It will be set by the put function or method performing the change. It cannot be set explicitly. |

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

### 2.42.4 Usage Conditions and Restrictions

2.42.4.1 The name will have to be a non-empty string with no white space or control characters.

2.42.4.2 The SiteSpecificChannelAttributes will be defined in the *Root* directory and shall be valid for all configurations.

## 2.43 Interface "ITypeDescriptions"

### 2.43.1 Description

2.43.1.1 This interface provides access to some reflection information. It shall allow COM clients of the Configuration Server to query available attributes of channel records.

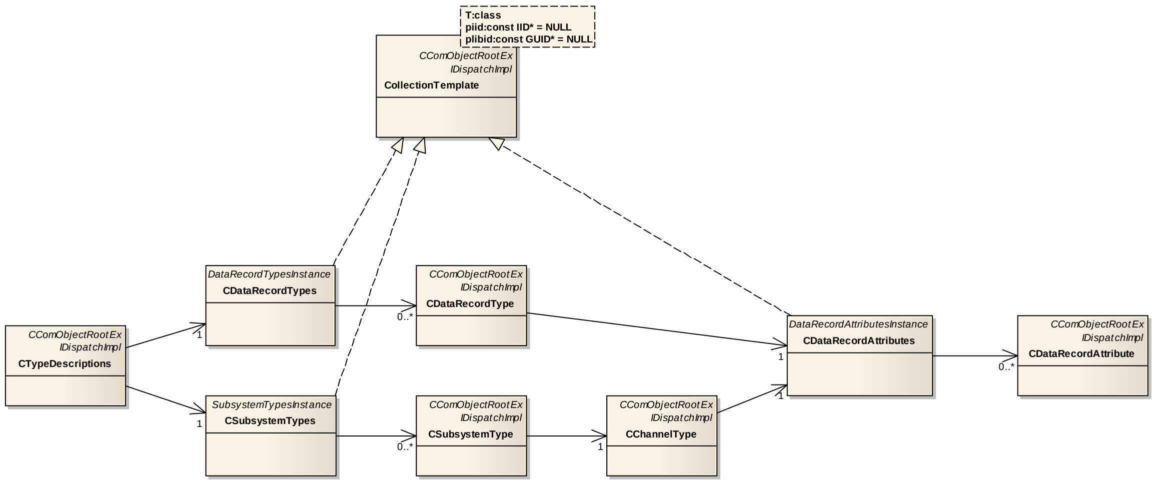

Figure 3: Class hierarchy for type descriptions

There are instances of the class CDataRecordType to describe the following types: IChannel, ISubsystem, IConfiguration, IEngineeringUnit, IMacro, ITransientLogDef, IBreakPointTable, IPolynomial. The instance for IChannel just describes the generic channel attributes. To get information about the subsystem-specific attributes, one has to use the CSubsystemType instances.

The instances of CChannelType describe the subsystem-specific attributes of channels of a subsystems.

### 2.43.2 Design

2.43.2.1 This interface shall be a dispatch interface.

2.43.2.2 This interface shall be an automation interface.

2.43.2.3 The properties of this interface are static – different instances of ITypeDescriptions will always return the same information.

### 2.43.3 Methods and Properties

2.43.3.1 Property DataRecordTypes

```
[propget, id(1)]
HRESULT DataRecordTypes([out, retval] LPDISPATCH *pVal)
```

| Argument Name | Description |
|---|---|
| pVal | Access to collection of description of data record types |

2.43.3.2 Property SubsystemTypes

```
[propget, id(2)]
HRESULT SubsystemTypes([out, retval] LPDISPATCH *pVal)
```

| Argument Name | Description |
|---|---|
| pVal | Access to collection of description of subsystem types |

## 2.44 Interface "IDataRecordTypes"

### 2.44.1 Description

2.44.1.1 This interface represents the collection of descriptions of data record types.

### 2.44.2 Design

2.44.2.1 This interface shall be a dispatch interface.

2.44.2.2 This interface shall be an automation interface.

2.44.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *IDataRecordType* interfaces in the collection.

2.44.2.4 This interface shall not be instantiated by the clients. An instance can be accessed by the *DataRecordTypes* property of the interface *ITypeDescriptions*.

### 2.44.3 Methods and Properties

2.44.3.1 Property Find

```
[id(1)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal)
```

| Argument Name | Description |
|---|---|
| Name | Name of data record type |
| pVal | Returned Description of data record type |

## 2.45 Interface "IDataRecordType"

### 2.45.1 Description

2.45.1.1 This interface represents a description of a data record type.

2.45.1.2 There are instances of the class CDataRecordType to describe the following types: IChannel, ISubsystem, IConfiguration, IEngineeringUnit, IMacro, ITransientLogDef, IBreakPointTable, IPolynomial.

2.45.1.3 Each instances provides information about the generic attributes of the described data record type.

### 2.45.2 Design

2.45.2.1 This interface shall be a dispatch interface.

2.45.2.2 This interface shall be an automation interface.

2.45.2.3 This interface shall not be instantiated by the clients. Instances can be accessed through the collection *IDataRecordTypes*.

### 2.45.3 Methods and Properties

2.45.3.1 Property Name

```
[propget, id(1)]
HRESULT Name([out, retval] BSTR *pVal)
```

| Argument Name | Description |
|---|---|
| pVal | Name of data record type |

2.45.3.2 Property Generic Attributes

```
[propget, id(2)]
HRESULT GenericAttributes([out, retval] LPDISPATCH *pVal)
```

| Argument Name | Description |
|---|---|

| | |
|---|---|
| pVal | Collection of descriptions of generic attributes of the data record type. |

## 2.46 Interface “IDataRecordAttributes”

### 2.46.1 Description

2.46.1.1 This interface represents the collection of descriptions of data record attributes.

E.g.: it contains a description of the “Name” attribute that is a property of each data record.

### 2.46.2 Design

2.46.2.1 This interface shall be a dispatch interface.

2.46.2.2 This interface shall be an automation interface.

2.46.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *IDataRecordAttribute* interfaces in the collection.

2.46.2.4 This interface shall not be instantiated by the clients. An instance can be accessed by the *GenericAttributes* property of the interface *IDataRecordType* or by the *SpecificAttributes* property of the interface *IChannelType*.

### 2.46.3 Methods and Properties

2.46.3.1 Property Find

```
[id(1)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal)
```

| **Argument Name** | **Description** |
|---|---|
| Name | Name of data record attribute |
| pVal | Returned Description of data record attribute |

## 2.47 Interface “IDataRecordAttribute”

### 2.47.1 Description

2.47.1.1 This interface represents a description of a data record attribute.

### 2.47.2 Design

2.47.2.1 This interface shall be a dispatch interface.

2.47.2.2 This interface shall be an automation interface.

2.47.2.3 This interface shall not be instantiated by the clients. Instances can be accessed through the collection *IDataRecordAttributes*.

### 2.47.3 Methods and Properties

2.47.3.1 Property Name

[propget, id(1)]
HRESULT Name([out, retval] BSTR *pVal)

| Argument Name | Description |
|---|---|
| pVal | Name of attribute |

2.47.3.2 Property DataType

[propget, id(2)]
HRESULT DataType ([out, retval] BSTR *pVal)

| Argument Name | Description |
|---|---|
| pVal | Type of values of the attribute. Following values are possible:<br>• STRING<br>• BOOL<br>• INTEGER<br>• DATETIME<br>• FLOAT<br>• ENUM |

## 2.48 Interface “ISubsystemTypes”

### 2.48.1 Description

2.48.1.1 This interface represents the collection of descriptions of subsystem types.

E.g. it contains a description for the subsystem type “Calculated”.

### 2.48.2 Design

2.48.2.1 This interface shall be a dispatch interface.

2.48.2.2 This interface shall be an automation interface.

Use or disclosure of the data on this sheet is subject to the restrictions on page i.

2.48.2.3 This interface shall implement the collection interface for *LPDISPATCH*, returning the *ISubsystemType* interfaces in the collection.

2.48.2.4 This interface shall not be instantiated by the clients. An instance can be accessed by the *SubsystemTypes* property of the interface *ITypeDescriptions*.

### 2.48.3 Methods and Properties

2.48.3.1 Property Find

```
[id(1)]
HRESULT Find([in] BSTR Name, [out, retval] long *pVal)
```

| **Argument Name** | **Description** |
|---|---|
| Name | Name of subsystem type |
| pVal | Returned Description of subsystem type |

## 2.49 Interface "ISubsystemType"

### 2.49.1 Description

2.49.1.1 This interface represents a description of a subsystem type.

### 2.49.2 Design

2.49.2.1 This interface shall be a dispatch interface.

2.49.2.2 This interface shall be an automation interface.

2.49.2.3 This interface shall not be instantiated by the clients. Instances can be accessed through the collection *ISubsystemTypes*.

### 2.49.3 Methods and Properties

2.49.3.1 Property Name

[propget, id(1)]
HRESULT Name([out, retval] BSTR *pVal)

| **Argument Name** | **Description** |
|---|---|
| pVal | Name of subsystem type |

2.49.3.2 Property ChannelType

[propget, id(2)]
HRESULT ChannelType([out, retval] LPDISPATCH *pVal)

| Argument Name | Description |
|---|---|
| pVal | Description of the channel type associated to this subsystem type |

## 2.50 Interface "IChannelType"

### 2.50.1 Description

2.50.1.1 This interface represents a description of a channel type.

E.g.: for the subsystem type "Calculated" it contains information about the specific channel attribute "Equation".

### 2.50.2 Design

2.50.2.1 This interface shall be a dispatch interface.

2.50.2.2 This interface shall be an automation interface.

2.50.2.3 This interface shall not be instantiated by the clients. Instances can be accessed through the interface *ISubsystemType*.

### 2.50.3 Methods and Properties

2.50.3.1 Property Name

[propget, id(1)]
HRESULT InterfaceName ([out, retval] BSTR *pVal);

| Argument Name | Description |
|---|---|
| pVal | Interface name of channel type |

2.50.3.2 Property SpecificAttributes

[propget, id(2)]
HRESULT SpecificAttributes ([out, retval] LPDISPATCH *pVal)

| Argument Name | Description |
|---|---|

| | |
|---|---|
| pVal | Collection of descriptions of specific attributes of channels of this type. |