OPC Batch Custom Interface Specification

Version 1.0

April 24, 2000

Specification Type	Industry Standard Specification		
Title:	**OPC Batch Custom Interface Specification**	Date:	April 24, 2000
Version:	1.0	Software:	MS-Word
		Source:	opcb10_cust.doc
Author:	Opc Foundation	Status:	**Released**

Synopsis:

This specification is the specification of the interface for developers of OPC clients and OPC servers. The specification is a result of an analysis and design process to develop a standard interface to facilitate the development of servers and clients by multiple vendors that shall inter-operate seamlessly together.

Trademarks:

Required Runtime Environment:

This specification requires Microsoft Windows 95, Windows 98, or Windows NT 4.0 or later

NON-EXCLUSIVE LICENSE AGREEMENT

The OPC Foundation, a non-profit corporation (the "OPC Foundation"), has established a set of standard OLE/COM interface protocols intended to foster greater interoperability between automation/control applications, field systems/devices, and business/office applications in the process control industry.

The current OPC specifications, prototype software examples and related documentation (collectively, the "OPC Materials"), form a set of standard OLE/COM interface protocols based upon the functional requirements of Microsoft's OLE/COM technology. Such technology defines standard objects, methods, and properties for servers of real-time information like distributed process systems, programmable logic controllers, smart field devices and analyzers in order to communicate the information that such servers contain to standard OLE/COM compliant technologies enabled devices (e.g., servers, applications, etc.).

The OPC Foundation will grant to you (the "User"), whether an individual or legal entity, a license to use, and provide User with a copy of, the current version of the OPC Materials so long as User abides by the terms contained in this Non-Exclusive License Agreement ("Agreement"). If User does not agree to the terms and conditions contained in this Agreement, the OPC Materials may not be used, and all copies (in all formats) of such materials in User's possession must either be destroyed or returned to the OPC Foundation. By using the OPC Materials, User (including any employees and agents of User) agrees to be bound by the terms of this Agreement.

LICENSE GRANT:

Subject to the terms and conditions of this Agreement, the OPC Foundation hereby grants to User a non-exclusive, royalty-free, limited license to use, copy, display and distribute the OPC Materials in order to make, use, sell or otherwise distribute any products and/or product literature that are compliant with the standards included in the OPC Materials.

All copies of the OPC Materials made and/or distributed by User must include all copyright and other proprietary rights notices include on or in the copy of such materials provided to User by the OPC Foundation.

The OPC Foundation shall retain all right, title and interest (including, without limitation, the copyrights) in the OPC Materials, subject to the limited license granted to User under this Agreement.

WARRANTY AND LIABILITY DISCLAIMERS:

User acknowledges that the OPC Foundation has provided the OPC Materials for informational purposes only in order to help User understand Microsoft's OLE/COM technology. THE OPC MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. USER BEARS ALL RISK RELATING TO QUALITY, DESIGN, USE AND PERFORMANCE OF THE OPC MATERIALS. The OPC Foundation and its members do not warrant that the OPC Materials, their design or their use will meet User's requirements, operate without interruption or be error free.

IN NO EVENT SHALL THE OPC FOUNDATION, ITS MEMBERS, OR ANY THIRD PARTY BE LIABLE FOR ANY COSTS, EXPENSES, LOSSES, DAMAGES (INCLUDING, BUT NOT LIMITED TO, DIRECT, INDIRECT, CONSEQUENTIAL, INCIDENTAL, SPECIAL OR PUNITIVE DAMAGES) OR INJURIES INCURRED BY USER OR ANY THIRD PARTY AS A RESULT OF THIS AGREEMENT OR ANY USE OF THE OPC MATERIALS.

GENERAL PROVISIONS:

This Agreement and User's license to the OPC Materials shall be terminated (a) by User ceasing all use of the OPC Materials, (b) by User obtaining a superseding version of the OPC Materials, or (c) by the OPC Foundation, at its option, if User commits a material breach hereof. Upon any termination of this Agreement, User shall immediately cease all use of the OPC Materials, destroy all copies thereof then in its possession and take such other actions as the OPC Foundation may reasonably request to ensure that no copies of the OPC Materials licensed under this Agreement remain in its possession.

User shall not export or re-export the OPC Materials or any product produced directly by the use thereof to any person or destination that is not authorized to receive them under the export control laws and regulations of the United States.

The Software and Documentation are provided with Restricted Rights. Use, duplication or disclosure by the U.S. government is subject to restrictions as set forth in (a) this Agreement pursuant to DFARs 227.7202-3(a); (b) subparagraph (c)(1)(i) of the Rights in Technical Data and Computer Software clause at DFARs 252.227-7013; or (c) the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 subdivision (c)(1) and (2), as applicable. Contractor/ manufacturer is the OPC Foundation, P.O. Box 140524, Austin, Texas 78714-0524.

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of the State of Minnesota, excluding its choice or law rules.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, the OPC Materials.

Table of Contents

1 INTRODUCTION 6

1.1 BACKGROUND 6
1.2 PURPOSE 6
1.3 REFERENCES 6
1.4 RELATIONSHIP TO OTHER OPC SPECIFICATIONS 6
1.5 SCOPE 7
1.6 TYPES OF BATCH SERVERS 7
1.7 AUDIENCE 8
1.8 DELIVERABLES 8

2 FUNDAMENTAL CONCEPTS 9

2.1 OVERVIEW 9
2.2 DATA SOURCES 10
2.3 GENERAL ARCHITECTURE AND COMPONENTS 11
2.4 OVERVIEW OF OBJECT AND INTERFACES 11

3 ARCHITECTURE 13

3.1 OVERVIEW 13
3.2 OPC BATCH NAMESPACE 16
3.2.1 Batch Namespace Models 16
3.2.2 Browsing the OPC Batch Namespace 18
3.2.2.1 Client Browsing Examples 19
3.2.3 Parameters and Results 22
3.2.3.1 Discovery of Parameters and Results 23
3.2.3.2 Anonymous Access of Parameters and Results 26
3.2.3.3 Parameter and Results Access Examples 28
3.2.4 Batch List 29
3.2.5 Handling Dynamic Data 30
3.2.6 Use of Delimiter 31
3.3 OPC BATCH PROPERTIES 32
3.3.1 Typical Use 32
3.3.2 How 'Property IDs' relate to ItemIDs 32
3.3.3 Property List 32
3.4 ENUMERATION CONCEPT 39
3.5 COMPLIANCE 42
3.6 OPC DATA ACCESS 42
3.7 RESERVED NAME SUMMARY 42

4 OPC BATCH SERVER CUSTOM INTERFACE QUICK REFERENCE 44

4.1 OPC BATCH SERVER OBJECT 45

5 OPC BATCH SERVER CUSTOM INTERFACES 46

5.1 OVERVIEW 46
5.2 OPCBATCHSERVER OBJECT 47
5.2.1 Overview 47
5.2.2 IUnknown 47
5.2.3 IOPCCommon 47
5.2.4 IOPCBatchServer 48
5.2.4.1 IOPCBatchServer::GetDelimiter 48
5.2.4.2 IOPCBatchServer::CreateEnumerator 48
5.2.5 IEnumOPCBatchSummary 50
5.2.5.1 IEnumOPCBatchSummary::Next 50
5.2.5.2 IEnumOPCBatchSummary::Skip 51

5.2.5.3 IEnumOPCBatchSummary::Reset 51
5.2.5.4 IEnumOPCBatchSummary::Clone 52
5.2.5.5 IEnumOPCBatchSummary::Count 52
5.2.6 IOPCEnumerationSets 53
5.2.6.1 IOPCEnumerationSets::QueryEnumerationSets 53
5.2.6.2 IOPCEnumerationSets::QueryEnumeration 54
5.2.6.3 IOPCEnumerationsSets::QueryEnumerationList 55

6 DESCRIPTION OF DATA TYPES, PARAMETERS AND STRUCTURES 56

6.1 STRUCTURES AND MASKS 56
6.1.1 OPCBATCHSUMMARY 56

7 INSTALLATION ISSUES 57

8 SUMMARY OF OPC ERROR CODES 58

9 APPENDIX A – OPC BATCH CUSTOM IDL SPECIFICATION 59

10 APPENDIX B OPCBATCHERROR.H 62

11 APPENDIX C – OPCBATCHDEF.H 63

12 APPENDIX D - OPCBATCHPROPS.H 65

1 Introduction

1.1 Background

As products are developed for the batch processing industry based on the IEC 61512-1 Batch Control – Part 1: Models and Terminology standard there is an increasing need to exchange data between these products and other systems. Interfaces occur at all levels; with Field Management devices (e.g. monitoring stations, control stations…), Process Management systems (e.g. lab systems, batch control systems, loading, unloading, dispensing, weighing systems…), and with Business Management systems (e.g. ERP and MES).

The data exchange covers four basic types of information such as equipment capabilities, current operating conditions, historical and recipe contents. Version 1.0 of this specification has been limited to equipment capabilities and current operating conditions. Equipment capabilities correspond to the IEC 61512-1 Physical Model. Current operating conditions are described using a Batch List and a Batch Model which combines Batch and Control Recipe information.

Currently most batch systems use their own proprietary interfaces for dissemination and collection of data. There is no capability to augment existing solutions with other capabilities in a plug-n-play environment. This requires the developer to recreate the same infrastructure for their products as all other vendors have had to develop independently with no interoperability with any other systems.

Manufacturers and consumers want to use off the shelf, open solutions from vendors that offer superior value that solves a specific need or problem.

1.2 Purpose

To provide a means to pass runtime batch and equipment data between components which would be suitable to standardization. Additionally this document details the design of interfaces and namespaces in such a way as to complement the existing OPC Data Access Interfaces.

1.3 References

- ?? OPC Data Access Custom Interface Specification, Version 2.0, OPC Foundation, October 13, 1998.
- ?? OPC Data Access Automation Interface Specification, Version 2.01, OPC Foundation, January 6, 1999.
- ?? OPC Common Definitions, Version 1.0, OPC Foundation, October 27, 1998.
- ?? IEC61512-1:1997, Batch control- Part 1: Models and terminology

1.4 Relationship to Other OPC Specifications

This specification differs from other OPC specifications in that it declares a well-defined namespace that must be supported. It is possible, and desirable, to declare a well-defined namespace in this case because of the existence of a widely accepted international standard that defines this namespace.

An OPC batch server must support all interfaces required by the OPC Data Access 2.0 specification as well as some of the optional interfaces.

1.5 Scope

This document represents the initial release of the batch specification. There are many areas of batch data exchange where an OPC standard interface would be beneficial. However, to achieve the goal of delivering the first interface within 24 months of the first meeting (the first meeting was March 1998) a smaller scope has been used for the first release.

The scope of the version 1.0 interface will deal with current runtime batch information and the equipment information required to understand the context of the runtime batch information.

Interfaces for master recipes, historical data, event, alarms, messages, prompts, scheduling data and other batch-related information have been delayed until a later specification. This was done to limit the size and complexity of the version 1.0 specification, not because the other data is unimportant.

The scope of this document is to provide a specification for a software "conduit" for runtime batch information to be read by clients from servers and optionally written by clients to servers. "Conduit" refers to the notion that this document is not intended to specify solutions for batch control problems, but rather provide an enabling technology that will permit multi-vendor solutions to operate in a heterogeneous computing environment.

1.6 Types of Batch Servers

It is expected that OPC batch servers will collaborate with execution engines for the process management and unit supervision activities in batch processing as defined in the Management Activity Model of IEC 61512-1 shown in figure 1-1. As later interfaces are added batch servers may then consist of systems performing recipe management, production planning and scheduling and production information management activities. However, no limitations are intended to limit the creation of batch servers that meet this specification.

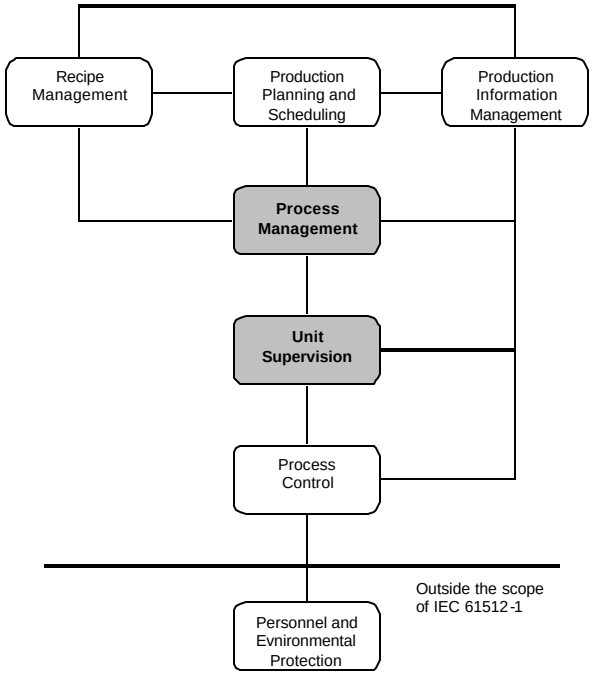


Figure 1-1 Management Activity Model[1]

[1] IEC 61512-1 Part 1: Batch control - Models and terminology, First Edition 1997-08.

1.7 Audience

This document is intended to be used as reference material for developers of OPC compliant batch clients and servers. It is assumed that the reader is familiar with Microsoft OLE/COM technology, the needs of the process control industry and the OPC Data Access 2.0 specification.

1.8 Deliverables

The deliverables from the OPC Foundation with respect to the OPC Batch Custom Interface Specification 1.0 include the OPC specification itself, OPC batch custom IDL files (included in this document as Appendices) and the OPC batch error header files (included in this document). As a convenience, standard proxystub DLLs and a standard batch header file for the OPC interfaces generated directly from the IDL will be provided at the OPC Foundation web site.

This OPC batch specification contains design information for the following:

1. **The OPC Batch Namespace** – A server independent syntax for accessing batch data.
2. **The OPC Batch Custom Interface** - This document will describe the interfaces and methods of OPC components and objects.

2 Fundamental Concepts

2.1 Overview

This specification describes the OPC COM objects and their interfaces implemented by OPC batch servers. An OPC Client can connect to OPC batch servers provided by one or more vendors. Different vendors may provide OPC batch servers. Vendor supplied code determines the data to which each server has access, the data names, and the details about how the server physically accesses that data. Vendors may also provide other OPC Servers along with their OPC batch server, but they are not required to. The following figure illustrates possible OPC vendor server configurations:

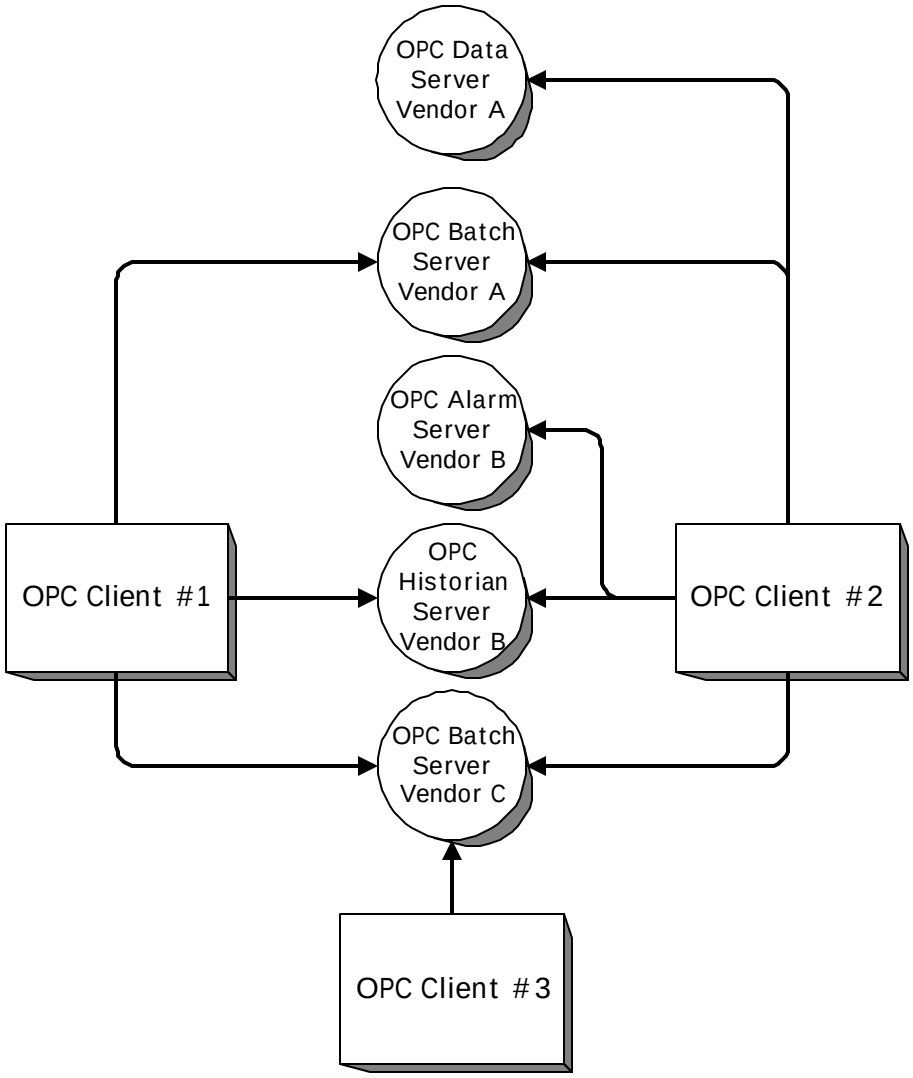


Figure 2-1: Server Interactions

A single client may interface with more than one server and/or more than one type of server. The clients and servers may be from the same or different vendors.

2.2 Data Sources

The OPC batch server provides a way to access or communicate to a set of batch data sources. The types of sources available are a function of the server implementation.

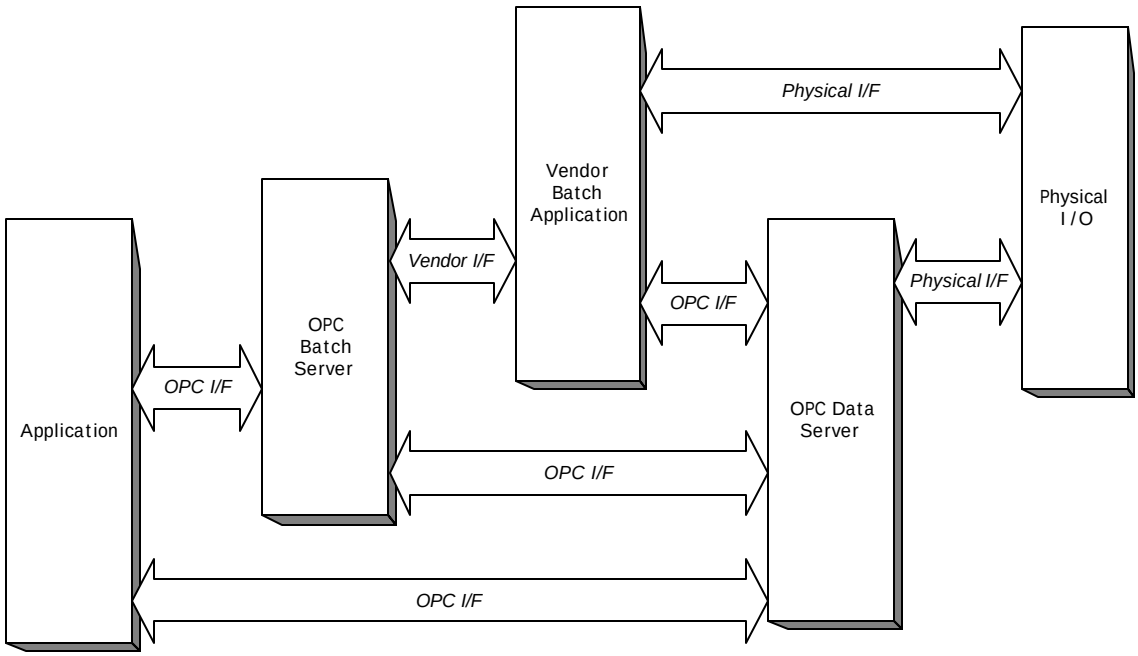


Figure 2-2: Possible OPC Batch Data Sources and Servers

The server may be implemented as a stand-alone OPC batch server that collects data from an OPC data access server or another data source. It may also be a set of interfaces that are layered on top of an existing Proprietary Batch Server. The clients that reference the OPC batch server may be simple applications that just want a few values or they may be complex displays or reports that require data in multiple formats.

2.3 General Architecture and components

An OPC client application communicates to an OPC batch server through the specified OPC custom and automation interfaces. OPC batch servers can implement both the custom interface and an automation interface.

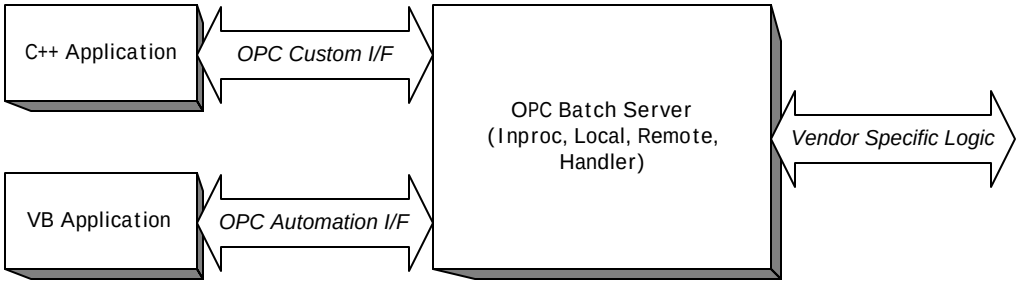


Figure 2-3: Custom and Automation Interfaces

The OPC Specification defines COM interfaces (what the interfaces are), not the implementation (not the how of the implementation) of those interfaces. It specifies the behavior that the interfaces are expected to provide to the client applications that use them.

Included are descriptions of architectures and interfaces that seemed most appropriate for those architectures. Like all COM implementations, the architecture of OPC is a client-server model where the OPC server component provides an interface to the OPC objects and manages them.

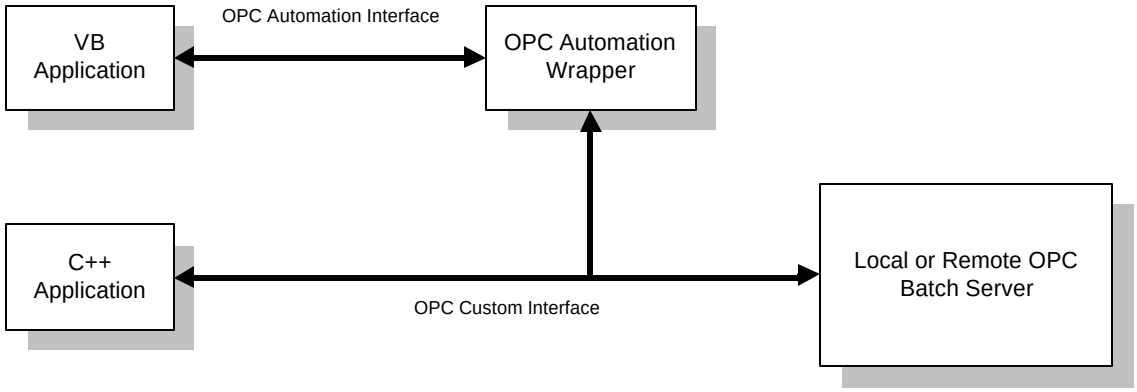


Figure 2-4: Automation Interface Wrapper

The OPC automation interface may be implemented via a wrapper. A generic wrapper will be provided by the OPC Foundation.

2.4 Overview of Object and Interfaces

The OPC batch server object provides the ability to read data from a batch server and to optionally write data to a batch server. It is acceptable for an OPC batch server to provide a read-only namespace. All COM objects are accessed

through interfaces. The client sees only the interfaces. Thus, the object described here is a 'logical' representation, which may not have anything to do with the actual internal implementation of the server. The following figure is a summary of the OPC batch server object and its interfaces. Interfaces with brackets are optional.

An OPC batch client must implement all of the Data Access 2.0 required client interfaces. There are no specific OPC batch client interfaces, however an OPC batch client must understand the OPC batch namespace.

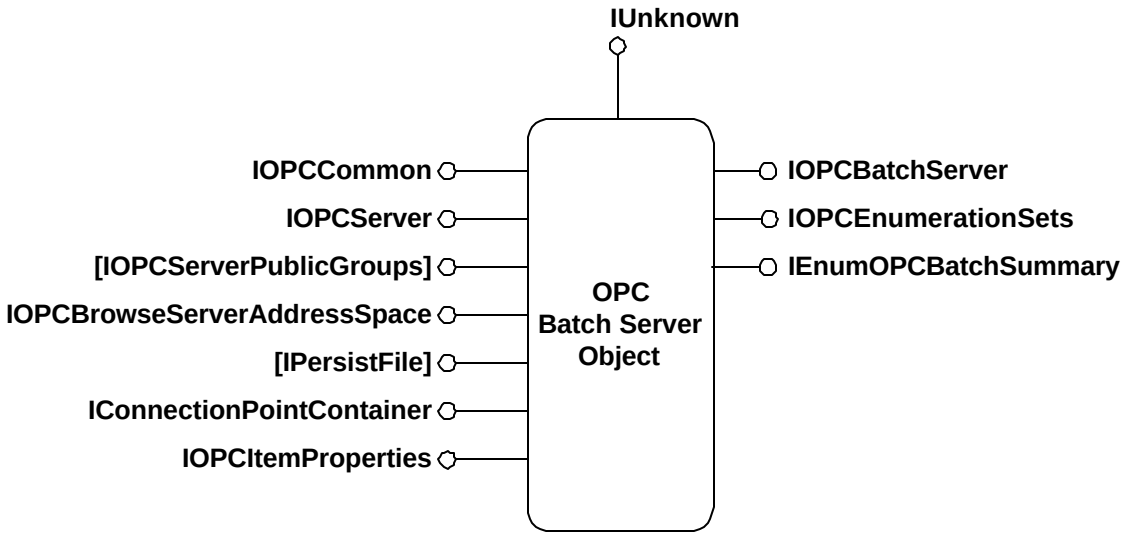


Figure 2-5- OPC Batch Server Object

3 Architecture

3.1 Overview

Version 1.0 of this specification builds upon other standards: The OPC Data Access Custom and Automation Interface Specifications 2.0 and the IEC 61512-1, Batch control – Part 1: Models and Terminology, First Edition 1997-08 standard. Although new interfaces are defined in this specification, the most important concept put forth is the implementation of the IEC 61512-1 models and terminology in a non-opaque, or well-known, namespace.

OPC data access servers have an opaque namespace, that is, using a general Item.Property syntax each client may obtain data from a server with few fixed requirements as to the "names" of the items and properties. For the most part the "names" are determined by the server. This is called an opaque namespace.

In the case of batch control, there exists a widely accepted standard that defines hierarchical models and terminology to support the models. The standard is IEC 61512-1 First Edition 1997-08. This is an international standard that has a U.S. counterpart in ANSI/ISA S88.01 1995. Both are titled "Batch control – Part 1: Models and terminology". Due to wide support by vendors and adoption by the batch processing industries there is a need to provide interoperability between batch control systems. This specification is a step in that direction.

Version 1.0 of this specification addresses runtime batch data related to the IEC 61512-1 Physical model and the Procedural control model. Runtime batch data has been interpreted to cover the hierarchical structure of a recipe and selected properties of control recipe elements. Later versions may add additional data, such as master recipes, procedure formats, alarms, events, prompts, and historical data to name a few.

The IEC 61512-1 Physical model shown in Figure 3-1 provides the OPC batch specification with the well-known items for an equipment hierarchy.

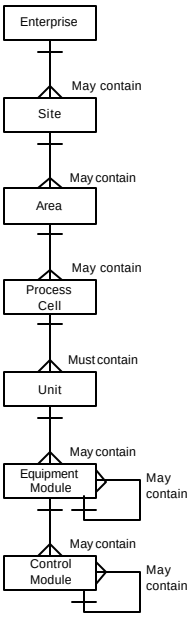


Figure 3-1 Physical model[2]

The IEC 61512-1 Procedural control module shown in Figure 3-2 defines the control recipe procedural elements. This model has been used to define the well-known item IDs inside a batch.

[2] IEC 61512-1 Part 1: Batch control - Models and terminology, First Edition 1997-08.

The OPC batch specification supports the concepts of collapsibility and expandability mentioned in IEC 61512-1. Collapsibility means that one or more levels in a model may be omitted for a specific implementation. Expandability means that additional levels may be inserted anywhere in the models. In the OPC batch interface each item (e.g. unit or equipment module) has a property called "ModelLevel". ModelLevel is given a value corresponding to the IEC 61512-1 term, or a user defined term. Using this technique levels may be omitted or new ones inserted in an implementation.

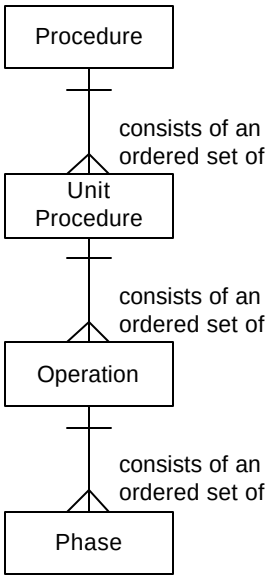


Figure 3-2 Procedural control module[3]

IEC 61512-1 defines a recipe as containing these five categories of information:

1. Header
2. Formula
3. Equipment Requirements
4. Procedure
5. Other Information.

Selected properties for all 5 categories are supported by version 1.0 of this specification as properties with well-known names. To make the organization of the properties as straightforward as possible properties for the five categories have been defined at all levels of the procedure hierarchy for control recipes. This encapsulation is shown in Figure 3-3.

If a category of data does not exist at a certain level in an implementation then the data will not be provided and an indication that no value exists will be returned.

[3] IEC 61512-1 Part 1: Batch control - Models and terminology, First Edition 1997-08.

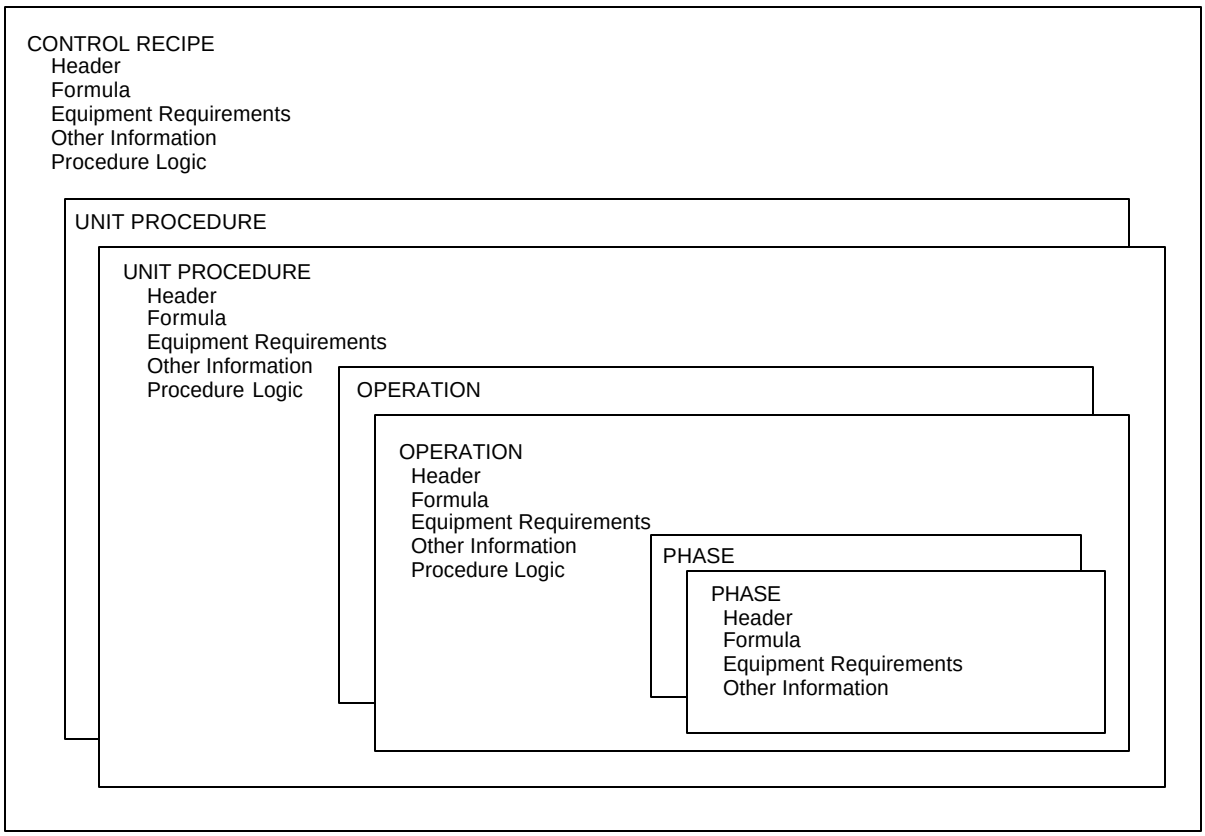


Figure 3-3: Recipe Element Encapsulation

IEC 61512-1 does not have a specific model to describe batch execution. To describe batch execution in this specification the "OPCBBatchModel" has been defined to describe the process management and unit supervision activities involved with control recipe execution. The Physical model provides the equipment environment in which batches execute. The Procedural model describes the hierarchy of procedural components in a control recipe. Together these models can be used to define the OPCBBatchModel. Figure 3-4 shows how the OPCBBatchModel is the union of the Physical and Procedural Models. The OPCBBatchModel is defined in the OPC Namespace section.

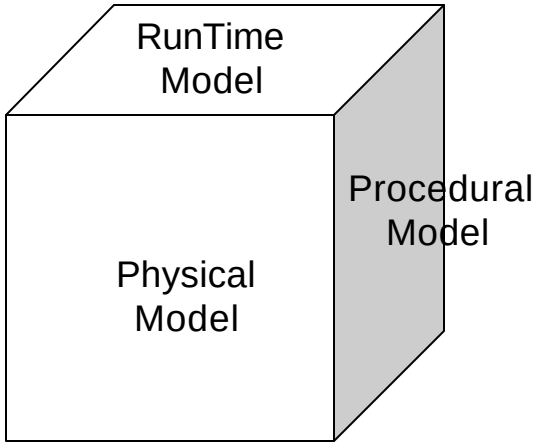
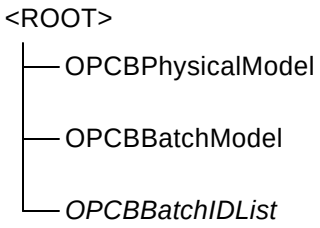


Figure 3-4 - Model Relationships

3.2 OPC Batch Namespace

3.2.1 Batch Namespace Models

The OPC batch namespace consists of a root and a series of well-known item IDs. All OPC batch well-known item IDs start with the pre-fix “OPCB”. "OPCB" is a reserved pre-fix and should not be used for vendor or user-defined item IDs. The well-known item IDs that exist immediately underneath the <ROOT> are OPCBPhysicalModel, OPCBBatchIDList, and OPCBBatchModel. They are called well-known item IDs since all OPC batch servers must support them in the OPC batch namespace. Clients may use the well-known item IDs to discover item IDs that have been grouped together and to access convenience functions. The OPC batch namespace is shown in Figure 3-5.



Note: Normal text indicates branches
Italics indicates leafs

Figure 3-5- OPC Batch Namesapce

The OPCBPhysicalModel well-known item ID is a branch that is used to contain the hierarchy of item IDs corresponding to the IEC 61512-1 Physical Model. The hierarchy under the physical model may have any number of levels. Each item ID has a property called OPCBPhysicalModelLevel (ID = 409) whose value identifies the model level (e.g. site, area, process cell, unit, equipment module, control module, or user defined levels). This method provides for expanding or collapsing the Physical Model. This means that users may create levels (e.g. train) or omit levels (e.g. site or area). This specification is only intended to provide a means to communicate a model, no attempt is made to enforce the IEC 61512-1 standard regarding the model structure.

The OPCBBatchModel well-known item ID is a branch that is used to contain the hierarchy of a batch and its control recipe. The hierarchy under the OPCBBatchModel may have any number of levels. Each item ID has a property called OPCBBatchModelLevel (ID = 410) whose value identifies the model level based on the IEC 61512-1 Procedure Model (e.g. procedure, unit procedure, operation, phase or user defined levels). As with the OPCBPhysicalModel hierarchy the OPCBBatchModel hierarchy provides for expanding and collapsing the Procedure Model. In this case users may expand the model by creating new recipe procedural elements (RPEs) such as a macro-operation, or omit levels, such as phases. Also as in the OPCBPhysicalModel hierarchy this means users may “mix-up” levels (e.g. a phase could contain a unit procedure). This specification does not encourage “mixing up” levels, but no attempt is made to enforce the standard model structure.

The OPCBBatchIDList well-known item ID is a leaf that contains information about a list of batches. The OPCBBatchIDList leaf returns a list of all the batch IDs the server has been configured to return to the client.

Each instance of an OPC batch namespace will have different hierarchies under the OPCBPhysicalModel and OPCBBatchModel. Therefore OPC batch clients will need to discover the namespace for each OPC batch server they connect with. Clients can discover a namespace by browsing it. For example a client may request a list of the batches a server is aware of. Using this list of batches the client may then request properties for a certain batch. If desired a client may browse down the procedure model for a batch to find information about a specific operation or phase.

Figure 3-6 shows an example of a populated OPC batch namespace. In this example the OPCBPhysicalModel hierarchy contains one site, named Site X, which contains one area, named Area 51, which contains two process cells named Building 19 ½ and Building 21. Each of the process cells contains two units. Building 19 ½ contains units Reactor 9 and Tank A, while Building 21 contains Reactor 10 and Tank B. This is a small example and is not carried down to the

equipment and control module levels to keep it small. However the hierarchy could contain any number of sites, areas, process cells, units, equipment modules or control modules. It could also omit any of these levels or add new levels.

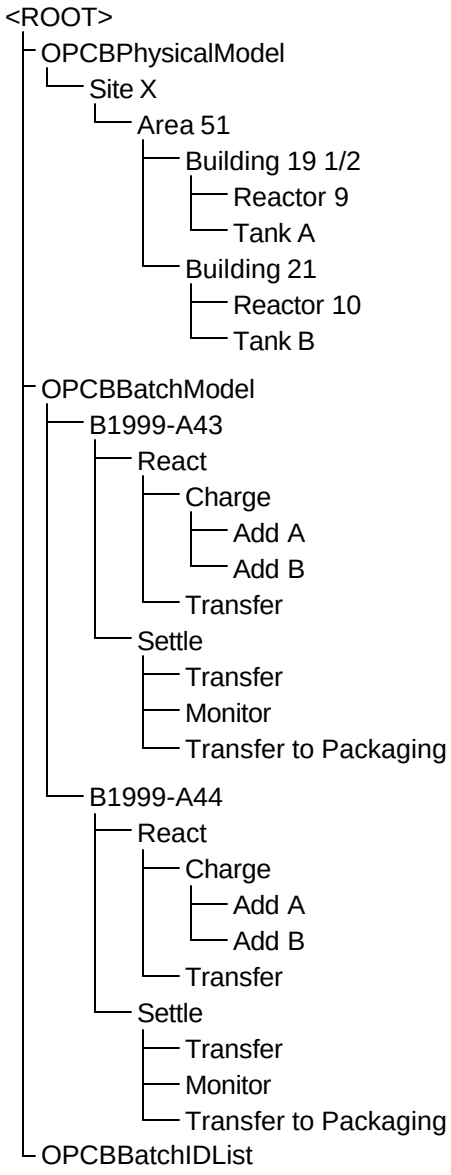


Figure 3-6- Populated Namespace

In this example the OPCBBatchModel hierarchy contains two batches, each derived from the same master recipe. B1999-A43 and B1999-A44 are the batch IDs of the two batches. Each of these batches contains two unit procedures, React and Settle. The React unit procedure contains two operations, Charge and Transfer. The Charge operation contains two phases, Add A and Add B. The Transfer operation does not contain any phases. The Settle unit procedure contains three operations, Transfer, Monitor and Transfer to Packaging, none of these operations contain phases.

Since the OPCBBatchIDList is a leaf there is no hierarchy shown below it. The OPCBBatchIDList would be accessed to obtain a list of the batch IDs that exists under the OPCBBatchModel (in this example, B1999-A43, B1999-A44).

The names used in the hierarchies under the OPCBPhysicalModel and OPCBBatchModel well-known item IDs may be found in either, or both, the value of the item ID or the value of the ID property (ID = 400) of the items. The ID property is intended to be used to identify an item, although its use is optional. An example of a returned value for the first batch in the OPCBBatchModel is:

B1999-A43

The term “fully qualified item ID” means that the value of the item ID is specific and unique enough to exactly identify the item. For example in the batch model when the same control recipe is used by more than one batch, identifying the unit procedure “React” is not sufficient to uniquely identify which instance of React is referenced. In this case the fully qualified item ID would be:

OPCBBatchModel.B1999-A43.React

When an OPC batch server returns an item ID it will always be a fully qualified item ID. In this example, and throughout this specification, a "." is used as the delimiter between item IDs. In practice the actual delimiter symbol is server specific. A method is provided in the batch server interface for clients to obtain the symbol used as the delimiter by a server.

The namespace may be accessed by either starting at the Root and browsing down or by using a fully qualified item ID to directly access an item. An example of a fully qualified item ID is:

OPCBPhysicalModel.Site X.Area 51.Building 19 1/2.Reactor 9

This item ID would permit a client to directly access properties for Reactor 9. Fully qualified item ID addressing may be used for any item in the OPC batch namespace.

3.2.2 Browsing the OPC Batch Namespace

The OPC batch namespace is hierarchical. A client may browse the namespace to discover what data is available for it to access. Browsing is performed using the IOPCBrowseServerAddressSpace interface, which is specified by the OPC Data Access Custom Interface Standard, version 2. In an OPC Data Access server this interface is optional, however, for an OPC batch server it is required.

In an OPC batch server, the namespace is always hierarchical, so the OPCNAMESPACETYPE returned from IOPCBrowseServerAddressSpace::QueryOrganization() will be OPC_NS_HIERARCHIAL (yes, opcda.h has this misspelled, but that’s life).

At any node of the hierarchy, two types of child nodes may be found. Branches are nodes that may have further branches or leaves under them. Leaves are nodes that represent single data items.

The OPC batch namespace is inherently dynamic. Batches, equipment or their components may be added or removed from the namespace under normal operating conditions. OPC batch clients must expect that the list of item IDs obtained from IOPCBrowseServerAddressSpace::BrowseOPCItemIDs() may change over time. Also the browse location of a client in the namespace may become invalid between data accesses (e.g. a batch is deleted from the server's namespace).

Review of IOPCBrowseServerAddressSpace

A full description of this interface may be found in the OPC Data Access Custom Interface Standard. The key points (from the Batch perspective) are reviewed here for convenience.

The important methods of IOPCBrowseServerAddressSpace for the purposes of the present discussion are:

ChangeBrowsePosition()	which moves the current browse position
BrowseOPCItemIDs()	which retums a list of child nodes found at the current position, using the COM interface IEnumString. Details of this interface may be found in the Microsoft documentation.
GetItemID()	returns an ItemID that may be used to access a data value

When a new IOPCBrowseServerAddressSpace interface pointer is obtained from the server, it will initially be set to the root of the hierarchical namespace. A call to IOPCBrowseServerAddressSpace::BrowseOPCItemIDs() with the filter type set to OPC_BRANCH returns the two branch nodes OPCBPhysicalModel and OPCBBatchModel which are defined by this specification.

From this point the client may call IOPCBrowseServerAddressSpace::ChangeBrowsePosition() to browse down to 'OPCBPhysicalModel' (for example). From there, another call to IOPCBrowseServerAddressSpace::BrowseOPCItemIDs() with the filter type set to OPC_BRANCH would return a list of Sites.

From any node, the client may call IOPCBrowseServerAddressSpace::BrowseOPCItemIDs() with the filter type set to OPC_LEAF. This returns an IEnumString with a list of any leaf nodes found. The entries returned in this list will not be fully qualified item IDs; they would simply be the names of the properties (such as "EquipmentClass"). To obtain a fully qualified item ID, the client must call IOPCBrowseServerAddressSpace::GetItemID(), passing the name of the leaf. This will return a string that may be used with the OPC Data Access methods (IOPCItemMgt::AddItems() etc.) to access the data.

When a client calls IOPCBrowseServerAddressSpace::BrowseOPCItemIDs() with the filter type set to OPC_FLAT the server returns strings which may be appended to the current browse position to create fully qualified item IDs for all items below the current position. For example if the browse position is:

OPCBPhysicalModel.Site X.Area 51

and a client requests the flat namespace below this position, then the following is returned:

Building 19 1/2

Building 19 1/2.Reactor 9

Building 19 1/2.Tank A

Building 21

Building 21.Reactor 10

Building 21.Tank B

3.2.2.1 Client Browsing Examples

Clients may initialize the browse position to the batch server <ROOT> or set browse position to a specific well-known position as in:

OPCBPhysicalModel.Site X.Area 51.Building 19 1/2.Reactor 9

If the current browse position is set to OPCBBatchModel, then when the browse interface is asked for a list of all branches, it will return a list of current batches (B1999-A43, B1999-A44 and B1999-A45). These nodes would be identified as branches, because there is always some RPE structure below the batch level.

The client could then browse down to the batch of interest at position OPCBBatchModel.B1999-A43. This position also represents an Item ID that potentially has a certain set of supported properties.

The client would call IOPCItemProperties::QueryAvailableProperties() to obtain a list of all the properties supported for this item ID and then either call IOPCItemProperties::GetItemProperties() to obtain a snapshot of the current property values, or call IOPCItemProperties::LookupItemIDs() for a list of desired properties and add the resulting item IDs to an OPC Group for periodic reading or subscription for change notification.

A client can use the information obtained from the browse interface, plus the item properties interface, to construct a list for the user that will allow them to select the properties for a given item ID. This list could be added to an OPC group. Table 1shows a sample list of fully qualified item IDs representing the properties of a batch; it is using the property name strings prescribed by the OPC Batch Custom Interface Specification. Note that this list could be much larger if vendor-specific properties were present.

Table 1 - Sample Properties for a Batch

[Batch Properties] -- for Batch B1999-A43
OPCBBatchModel.B1999-A43.ID
OPCBBatchModel.B1999-A43.Value
OPCBBatchModel.B1999-A43.AccessRights
OPCBBatchModel.B1999-A43.EU
OPCBBatchModel.B1999-A43.Description
OPCBBatchModel.B1999-A43.HighValueLimit
OPCBBatchModel.B1999-A43.LowValueLimit
OPCBBatchModel.B1999-A43.TimeZone
OPCBBatchModel.B1999-A43.ConditionStatus
OPCBBatchModel.B1999-A43.OPCBBatchModelLevel
OPCBBatchModel.B1999-A43.Version
OPCBBatchModel.B1999-A43.AllocatedEquipmentList
OPCBBatchModel.B1999-A43.RequesterList
OPCBBatchModel.B1999-A43.RequestedList
OPCBBatchModel.B1999-A43.SharedByList
OPCBBatchModel.B1999-A43.CampaignID
OPCBBatchModel.B1999-A43.LotIDList
OPCBBatchModel.B1999-A43.ControlRecipeID
OPCBBatchModel.B1999-A43.ControlRecipeVersion
OPCBBatchModel.B1999-A43.MasterRecipeID
OPCBBatchModel.B1999-A43.MasterRecipeVersion
OPCBBatchModel.B1999-A43.ProductID
OPCBBatchModel.B1999-A43.Grade
OPCBBatchModel.B1999-A43.BatchSize
OPCBBatchModel.B1999-A43.Priority
OPCBBatchModel.B1999-A43.ExecutionState
OPCBBatchModel.B1999-A43.IEC61512-1State
OPCBBatchModel.B1999-A43.ExecutionMode
OPCBBatchModel.B1999-A43.IEC61512-1Mode
OPCBBatchModel.B1999-A43.ScheduleStartTime
OPCBBatchModel.B1999-A43.ActualStartTime
OPCBBatchModel.B1999-A43.EstimatedEndTime
OPCBBatchModel.B1999-A43.ActualEndTime
OPCBBatchModel.B1999-A43.OPCBPhysicalModelReference
OPCBBatchModel.B1999-A43.EquipmentProceduralElement
OPCBBatchModel.B1999-A43.ParameterCount
OPCBBatchModel.B1999-A43.Parameter.Type
OPCBBatchModel.B1999-A43.ValidValues
OPCBBatchModel.B1999-A43.ScalingRule
OPCBBatchModel.B1999-A43.ExpressionRule
OPCBBatchModel.B1999-A43.ResultCount
OPCBBatchModel.B1999-A43.EnumerationSetID

Once an item ID like the ones above has been obtained, for example, OPCBBatchModel.B1999-A43.ProductID**,** this item ID is like any other item ID and may be added to an OPC group and then either read synchronously or subscribed to for change notification.

A client at the browse position OPCBBatchModel.B1999-A43 can ask the server to show the branch nodes at this level. The browse interface would return a list containing the unit procedure RPEs named React and Settle, and optionally two

additional branch names, OPCBParameters and OPCBResults, which are discussed in section 3.2.3 Parameters and Results. Only two unit procedures are used in this example for simplicity, however a namespace may contain any number of RPEs at this level, in which case the list would be longer.

Using IOPCBrowseServerAddressSpace::GetItemID(), the qualified Item ID OPCBBatchModel.B1999-A43.React would be obtained. This position also represents an item ID that potentially has a set of properties. The client would call IOPCItemProperties::QueryAvailableProperties() to obtain a list of all the properties supported for this item ID and then either call IOPCItemProperties::GetItemProperties() to obtain a snapshot of the current property values or call IOPCItemProperties::LookupItemIDs() for a list of desired properties and add the resulting item IDs to an OPC group for periodic reading or subscription for change notification.

As with browsing at the batch level, the property name strings defined in the OPC Batch Custom Specification may be appended to the base item ID and used to obtain property values for an item ID. When this is done the resulting string is itself a fully qualified item ID. When fully qualified item IDs are constructed the client is responsible for using the correct delimiter (see IOPCBatchServer::GetDelimiter). A sample list of item IDs representing the properties of the React unit procedure is shown in Table 2.

Table 2 - Sample Properties for RPEs Below the Batch Model Level

[RPE Properties] – for the RPE React, a Unit Procedure in Batch B1999-A43
OPCBBatchModel.B1999-A43.React.ID
OPCBBatchModel.B1999-A43.React.Value
OPCBBatchModel.B1999-A43.React.AccessRights
OPCBBatchModel.B1999-A43.React.EU
OPCBBatchModel.B1999-A43.React.Description
OPCBBatchModel.B1999-A43.React.HighValueLimit
OPCBBatchModel.B1999-A43.React.LowValueLimit
OPCBBatchModel.B1999-A43.React.TimeZone
OPCBBatchModel.B1999-A43.React.ConditionStatus
OPCBBatchModel.B1999-A43.React.OPCBBatchModelLevel
OPCBBatchModel.B1999-A43.React.Version
OPCBBatchModel.B1999-A43.React.AllocatedEquipmentList
OPCBBatchModel.B1999-A43.React.RequesterList
OPCBBatchModel.B1999-A43.React.RequestedList
OPCBBatchModel.B1999-A43.React.SharedByList
OPCBBatchModel.B1999-A43.React.LotIDList
OPCBBatchModel.B1999-A43.React.ExecutionState
OPCBBatchModel.B1999-A43.React.IEC61512-1State
OPCBBatchModel.B1999-A43.React.ExecutionMode
OPCBBatchModel.B1999-A43.React.IEC61512-1Mode
OPCBBatchModel.B1999-A43.React.ScheduleStartTime
OPCBBatchModel.B1999-A43.React.ActualStartTime
OPCBBatchModel.B1999-A43.React.Estimated EndTime
OPCBBatchModel.B1999-A43.React.ActualEndTime
OPCBBatchModel.B1999-A43.React.EquipmentProceduralElement
OPCBBatchModel.B1999-A43.React.ParameterCount
OPCBBatchModel.B1999-A43.React.ParameterType
OPCBBatchModel.B1999-A43.React.ValidValues
OPCBBatchModel.B1999-A43.React.ScalingRule
OPCBBatchModel.B1999-A43.React.ExpressionRule
OPCBBatchModel.B1999-A43.React.ResultCount
OPCBBatchModel.B1999-A43.React.EnumerationSetID
OPCBBatchModel.B1999-A43.React.OPCBParameters
OPCBBatchModel.B1999-A43.React.OPCBResults

In this example the RPE property OPCBBatchModel.B1999-A43.ParameterCount allows the client to determine if the React unit procedure has any parameters without browsing down into the OPCBParameters branch. Whether this is read

as a property or as a simple item ID via OPC Data Access, ParameterCount is a required Item Property at all RPE levels below the batch level. The corresponding ResultCount property serves the same purpose for result data.

If a particular RPE level does not have any parameters (or results) associated with it, then the returned value for ParameterCount (or ResultCount) is zero (0). It is also permissible for a server browse interface NOT to return the branch called OPCBParameters if the ParameterCount is 0.

A client may use the ParameterCount (and ResultCount) property to determine how to handle access to the parameters (and results). If there are a few or dozens, a client may use one approach; if there are hundreds or thousands, a client may choose to take another approach or warn the user of possible performance considerations.

3.2.3 Parameters and Results

Parameters and results represent a class of information that may be associated with batches and/or recipe procedural elements at any level in the OPCBBatchModel.

Parameters are used to send data values from a recipe procedural element to an associated equipment procedural element (EPE). Results are used to send actual values resulting from an execution of an RPE in an associated EPE to some other user of batch information, such as the production management system or a production information management system. Parameters and results are handled in an identical manner, except that results have fewer properties than parameters do.

Sets of parameters and results may occur at any level in the OPCBBatchModel hierarchy. A set of parameters and results for a given RPE may range from none, to one, to hundreds of individual parameters and results, each with a possibly large set of supported standard item properties and additional vendor-specific properties. Thus, the amount of information in a batch server can be very large and this data must be served to a potential set of many clients, each of which may be accessing some or all of the available information.

Some clients may wish to download all of the available parameter information at startup and then track real-time changes, other clients may poll a specific set of parameters and watch for changes in a specific set of results. There needs to be a range of access mechanisms to allow each type of client to efficiently do its job.

Table 3 contains a fragmentary namespace for an OPC batch server. Note that the parameters and results for any given RPE are shown separately and in curly braces {} to indicate that how access is gained to that information is not fully described at this point. Rather, this example simply highlights that parameter and result information, if present, are logically nested below their corresponding RPE node in the namespace.

Table 3 - Batch Model Branch Skeleton

<ROOT>	
OPCBPhysicalModel	Top of the equipment hierarchy.
…	
OPCBBatchModel	Top of the batch model hierarchy that contains batches the OPC batch server knows about at this time.
B1999-A43	One of the batches the batch server knows about.
[Batch Properties]	Item properties of this Batch – see Table 1.
{OPCBParameters/OPCBResults}	The parameter and result information we want to access – see Table 4.
React	A unit procedure.
[RPE Properties]	Properties for the unit procedure React (see Table 2).
{OPCBParameters/OPCBResults}	Parameter and/or result data for RPE (unit procedure) React in Batch B19999-A43.
DoubleCharge	An operation.
[RPE Properties]	Properties for the operation DoubleCharge.
{OPCBParameters/OPCBResults}	Parameter and/or result data for the RPE (operation) DoubleCharge in the React unit procedure.
ChargeA	A phase.
[RPE Properties]	Properties for the phase ChargeA.
{Parameters/OPCBResults}	Parameter and/or result data for the RPE (phase) ChargeA.

ChargeB	Another phase in the operation DoubleCharge.
[RPE Properties]	Properties for the phase ChargeB.
{Parameters/OPCBResults}	for RPE (Phase) ChargeA in DoubleCharge
B1999-A44	Another Batch.
B1999-A45	Yet another Batch.
OPCBBatchIDList	IDs of Batches this OPC batch server knows about at this time

3.2.3.1 Discovery of Parameters and Results

One method for a client to determine the parameters associated with a batch or RPE is to discover them. Browsing down the namespace using well-known item IDs does this. The well-known item IDs are called OPCBParameters and OPCBResults. They are collection points for parameter and result item IDs. Every batch and RPE item ID will have these well-known item IDs if there are parameter and/or result item IDs associated with them.

The namespace example from Table 3 has been expanded in Table 4 to show the OPCBParameters and OPCBResults well-known item IDs. In this example the parameter item IDs (e.g. Catalyst_Quantity, Ramp_Rate, and Heating_Duration) for the unit procedure with the fully qualified item ID OPCBBatchModel.B1999-A43.React would be found below the item ID OPCBBatchModel.B1999-A43.React.OPCBParameters. If a client with the browse position OPCBBatchModel.B1999-A43.React requests a list of branches, then OPCBParameters and OPCBResults will be returned along with the item ID for the RPE (operation) called DoubleCharge.

Table 4 - Parameters and Results Location in the RPE Namespace

B1999-A43	One of the batches this batch server knows about.
[Batch Properties]	Item properties of this Batch.
OPCBParameters	
OPCBResults	
React	A unit procedure.
[RPE Properties]	Properties for the React unit procedure.
OPCBParameters	
OPCBResults	
DoubleCharge	An operation.
[RPE Properties]	Properties for the operation DoubleCharge.
OPCBParameters	
OPCBResults	
ChargeA	A phase.
[RPE Properties]	Properties for the phase ChargeA.

The discovery method permits a client to gain access to all the parameters (each of which is represented as an item ID) for a batch or RPE item ID. Likewise with OPCBResults a user may reference a path like OPCBBatchModel.B1999-A43.React.OPCBResults to access all the result item IDs for the unit procedure React. A client would be able to move to the branch, DoubleCharge, and continue down the RPE tree or it would be able to move to the branch OPCBParameters to find out more about the parameters for React. Once the client has reached a particular RPE branch and finds that the branches OPCBParameters and OPCBResults exist, it is possible to browse to the next level to see all the individual parameters and results.

Continuing the example from Table 3 and Table 4, Table 5 shows the RPE React, its three parameters and two results. In the hierarchy directly below React are all of its properties, the well-known item IDs OPCBParameters and OPCBResults as branches, and the branch item ID representing the operation DoubleCharge.

Below the OPCBParameters and OPCBResults branches are lower-level branches, one branch for each supported parameter or result.

Table 5 - Drill Down to Parameters and Results

B1999-A43	One of the Batches this Batch Server knows about.
React	A unit procedure.
[RPE Properties]	Properties for React.
OPCBParameters	Well-known item ID branch under which is a collection of item IDs representing parameters associated with React.
[Parameter Collection Properties]	Properties of the well-known item ID OPCBParameters.
Catalyst_Quantity	Parameter for the unit procedure React.
[Parameter Properties]	Properties for the parameter Catalyst_Quantity (i.e.. ID = "Catalyst_Quantity" and OPCBBatchModelLevel = 5 which corresponds to the enumeration OPCB_PROC_PARAMETER).
Ramp_Rate	Parameter for the unit procedure React.
[Parameter Properties]	Properties for the parameter Ramp_Rate.
Heating_Duration	Parameter for the unit procedure React.
[Parameter Properties]	Properties for the parameter Heating_Duration.
OPCBResults	Well-known item ID branch under which is a collection of item IDs representing results associated with React.
[Result Collection Properties]	Properties of the well-known item ID OPCBResults.
Reaction_Duration	Result for the unit procedure React.
[Result Properties]	Properties for the result Reaction_Duration (i.e.. ID = "Reaction_Duration" and OPCBBatchModelLevel = 7 which corresponds to the enumeration OPCB_PROC_RESULT).
DoubleCharge	Item ID branch for the operation DoubleCharge that is part of the unit procedure React.

The well-known item IDs representing parameter and result collections must have the same set of properties as the parameters and results within the collection. Therefore a client may obtain the set of properties for all the associated parameters or results for an item in the OPCBBatchModel by calling IOPCItemProperties::QueryAvailableProperties() on the OPCBParameters or OPCBResults items respectively. The OPCBBatchModelLevel property will identify the well-known item ID as an OPCB_PROC_PARAMETER_COLLECTION or an OPC_PROC_RESULT_COLLECTION.

Each parameter and result has a set of properties associated with it. The OPCBBatchModelLevel property will identify the parameter item IDs as OPCB_PROC_PARAMETER and the result item IDs as OPCB_PROC_RESULT. Parameter and result item IDs are a subset of properties defined in section 3.3, OPC Batch Properties as well as possibly having vendor defined properties.

Sample Parameter and Result Properties

Use of the OPCBParameters and OPCBResults well-known item IDs permits clients to browse through the parameters and results associated with each batch and/or RPE. Browsing permits clients to access all the parameters and results by name, where the name is both the value of the well-known item ID and the ID property.

Table 6 contains a set of sample property item IDs for the Catalyst_Quantity parameter from Table 5. These fully qualified item IDs could be constructed by browsing down into the OPCBParameters branch under the unit procedure React and then combining this item ID with the item properties.

Table 6 - Sample Properties for Parameters

OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.ID
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.Value
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.AccessRights
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.EU
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.Description
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.HighValueLimit
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.LowValueLimit
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.Timezone
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity. OPCBBatchModelLevel
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.ParameterType
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.ValidValues
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.ScalingRule
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.ExpressionRule
OPCBBatchModel.B1999-A43.React.OPCBParameters.Catalyst_Quantity.EnumerationSetID

The same technique could be used for results. An example is shown in Table 7.

Table 7 - Sample Properties for Results

OPCBBatchModel.B1999-A43.React.OPCBResults.Reaction_Duration.ID
OPCBBatchModel.B1999-A43.React.OPCBResults.Reaction_Duration.Value
OPCBBatchModel.B1999-A43.React.OPCBResults.Reaction_Duration.AccessRights
OPCBBatchModel.B1999-A43.React.OPCBResults.Reaction_Duration.EU
OPCBBatchModel.B1999-A43.React.OPCBResults.Reaction_Duration.Description
OPCBBatchModel.B1999-A43.React.OPCBResults.Reaction_Duration.Timezone
OPCBBatchModel.B1999-A43.React.OPCBResults.Reaction_Duration.OPCBBatchModelLevel
OPCBBatchModel.B1999-A43.React.OPCBResults.Reaction_Duration.EnumerationSetID

3.2.3.2 Anonymous Access of Parameters and Results

Discovering each parameter and result item ID for an RPE may not always be desirable. An alternative method for obtaining parameter and result data is to do so anonymously. Anonymous access of parameters and results is a shorthand notation that provides clients with the ability to access parameter / result data without undertaking the discovery process.

Once a client has discovered an RPE and found a non-zero value associated with ParameterCount and/or ResultCount, a client could append the following to the RPE's fully qualified item ID to obtain parameter properties:

.OPCBParameters.OPCBPn.{property name}

.OPCBResults.OPCBRn.{property name}

where:

OPCBParameters	Well-known item ID for the collection of parameters for a batch or RPE.
OPCBResults	Well-known item ID for the collection of results for a batch or RPE.
OPCBPn	Anonymous parameter access, where n represents the 1-based index into the parameter array and has an upper bound equal to the value of the ParameterCount property for the RPE.
OPCBRn	Anonymous result access, where n represents the 1-based index into the result array and has an upper bound equal to the value of the ResultCount property for the RPE.
{propertyname}	Represents the desired property associated with the parameter or result (e.g., ID, Value, EU, Description…).

For applications wishing to simply display the required properties as well as the parameter / result data, the client program can simply use the browse model to discover RPE levels (i.e., branches) and construct Item IDs, by appending property names as well as the shorthand notation described above.

For example, a client that has discovered the following RPE:

OPCBBatchModel.B1999-A43.React

may construct the following Item IDs:

OPCBBatchModel.B1999-A43.React.ID
OPCBBatchModel.B1999-A43.React.Value
OPCBBatchModel.B1999-A43.React.ParameterCount
OPCBBatchModel.B1999-A43.React.ResultCount

and once the value for ParameterCount have been read, can construct:

OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP1.ID
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP1.Value

OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP2.ID
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP2.Value

OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP3..ID
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP3.Value

OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP4.ID
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP4.Value

A similar technique can be used for anonymously accessing results.

This technique can be used to obtain any properties for all of the parameters and/or results for a batch or RPE without having to first discover all of the parameter IDs.

For applications wishing to access more than just the required properties for parameters / results, the use of the IOPCItemProperties or the IOPCBrowseServerAddressSpace interfaces can be employed using these shorthand constructed Item IDs.

Shorthand notation for parameters / results adheres to the following:

?? Constructed Item IDs using the reserved strings OPCBPn and OPCBRn exist for every parameter and result defined in the namespace.

?? A constructed fully qualified Item ID is valid wherever a method accepts Item IDs as a parameter (e.g IOPCItemProperties::QueryAvailableProperties()).

?? The shorthand notation is an alias to a valid item ID and is never returned from the server.

3.2.3.3 Parameter and Results Access Examples

There are many possibilities for accessing parameter and result data. The following examples illustrate only a couple of different approaches and are not intended to impose a particular design or implementation on client applications. For the sake of brevity, the following examples are simply summarized instead of providing actual code examples.

Example 1

A client simply wishes to display the Value and Description properties for the parameters associated with a discovered RPE.

The client application first reads the ParameterCount property for the RPE to determine the number of parameters for the associated RPE. The client application then constructs a list of new Item IDs, by concatenating the RPE Item ID with the string ".OPCBPn.Value" where n is replaced with integer values from 1 to the number identified by ParameterCount. The client also constructs in a similar manner a list using ".OPCBPn.Description". These constructed Item IDs are then added to an OPC Group and a read operation is performed.

Example 2

A client wishes to display all of the properties for each of the defined parameters associated with a discovered RPE.

Since the client wishes to access all properties for the parameters, the client must first discover which properties exist for each of the parameters. Note that each parameter may have a different set of properties associated with it, so simply discovering the supported properties for one parameter does not imply anything about the properties for another.

Property discovery can be accomplished in two ways. One using the IOPCBrowseServerAddressSpace interface and the other using the IOPCItemProperties interface.

Using the IOPCBrowseServerAddressSpace interface, the client would browse into the OPCBParameters branch, retrieving the parameter names, then likewise browsing into each parameter to discover the supported properties, and finally adding the Item IDs for the discovered properties to an OPC Group.

Using the IOPCItemProperties interface, the client could construct a list of Item IDs by appending the string ".OPCBPn" where n represents a parameter index from 1 to ParameterCount for each of the defined parameters. The client would then query (using IOPCItemProperties::QueryAvailableProperties) the supported properties and read their values (using IOCPItemProperties::GetItemProperties) for each of the constructed Item IDs. Note that while this specific example used the ParameterCount to construct the item ID passed to the IOPCItemProperties interface, it could also have used the IOPCBrowseServerAddressSpace interface to determine the parameter names and use these to pass into the IOPCItemProperties interface. Table 8 provides shows the fully qualified item IDs for anonymous access.

Table 8 - Accessing Common Properties Anonymously

OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP1.ID
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP1.Value
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP1.AccessRights
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP1.EU
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP1.Description
.
.
.
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP29.ID
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP29.Value
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP29.AccessRights
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP29.EU
OPCBBatchModel.B1999-A43.React.OPCBParameters.OPCBP29.Description

3.2.4 Batch List

Interaction with a batch system is most often viewed through the context of what is currently executing in the batch system;

- ?? what is the list of batches being processed,
- ?? waiting to be processed, or
- ?? have completed processing.

The list of batches may be viewed as a simple list of batch IDs or a more detailed list that includes such attributes as state and recipe ID. Both of these views of the list of batches are important depending upon the usage of the information.

For simple and ad-hoc HMI interfaces, the needs of the operator are to have a summary of each batch known in the batch system. The summary has been implemented using the OPCBATCHSUMMARY structure.

For more complex application interfaces, the need to access the list of batches is geared to retrieving significant batch list data as a means to access other parts of the recipe procedural elements within a batch system. For example, a client application may wish to provide the means to the operator to drill down into the batch to see the individual recipe components. These two methods require two distinct access methods.

The first method uses OPC Data Access Specification 2.0 to access the list of batches through the well-known item ID "OPCBBatchIDList". This item ID returns a list of OPC item IDs representing the batches known to the batch system. The OPCBBatchIDList item ID value is VT_ARRAY of VT_BSTR.

From these item IDs, the client can retrieve specific properties for each batch by combining the item ID and the defined properties for the Batch object. For example, the Batch List returns 2 entries, with the following item IDs:

OPC Item ID for 1st batch - "B1999-A43"

OPC Item ID for 2nd batch - "B1999-A44"

To retrieve the State of the 1st batch, the client application would construct the following OPC item ID:

"OPCBBatchModel.B1999-A43.State".

Note that the OPCBBatchIDList does not return fully qualified item IDs.

The second method defines an enumeration interface, IEnumOPCBatchSummary that allows client applications the ability to retrieve the batch list with little overhead. This interface provides the client the ability to retrieve the batch list in a single read and furthermore it serves as the basis for the automation interface.

3.2.5 Handling Dynamic Data

Some batch data is inherently dynamic in the sense that it may appear and disappear when, for example, batches are added to or removed from the batch list. In OPC Data Access terms, this means that item IDs may cease to be valid outside of the client's control.

This creates the need to define behavior of the existing OPC Data Access interfaces, so servers may predictably notify clients when this occurs. The OPC batch specification does not define any new return codes over those defined in OPC Data Access Specification 2.0, but clarifies the usage of some existing return codes in order to handle this dynamic data.

Invalid Current Browse Position

If the current browse position has become invalid all calls dependent upon browse position return E_FAIL.

If the current browse position has become invalid, the client has the option to employ various strategies for error recovery. The simplest option is to use IOPCBrowseServerAddressSpace::ChangeBrowsePosition() to return to the root and restart browsing from there (this would be a rather disruptive for an interactive user, but might be workable for a client that is browsing programmatically). Note to reset the browse position to the root a pointer to a null string is used per the OPC Data Access 2.0 Specification section 4.4.8.

A more sophisticated approach would be for the client to remove the trailing portion of the item ID that represents the current browse position, and use IOPCBrowseServerAddressSpace::ChangeBrowsePosition() to try to browse to that position and try to restart browsing from there. Conceptually, this is moving up one level in the hierarchy and checking to see if it is valid.

IOPCBrowseServerAddressSpace::ChangeBrowsePosition

If the szString parameter to this method contains a string that does not correspond to a valid branch, E_INVALIDARG is returned. This could happen either if the string represented a branch that had disappeared due to its dynamic nature, or if it was simply incorrect (had never existed). Note that E_INVALIDARG might also be returned in other error situations.

IOPCBrowseServerAddressSpace::BrowseOPCItemIDs

Clients should be aware that successive calls to BrowseOPCItemIDs might return different information. The enumerator returned as ppIEnumString may contain a different number of elements on each call.

IOPCBrowseServerAddressSpace::GetItemID

Clients should be aware that a branch name that was previously returned as valid from BrowseOPCItemIDs might at a later time not be valid. GetItemID will return E_INVALIDARG in this case.

Handling Dynamic Data in other OPC Data Access methods

An OPC batch server is also an OPC Data Access Server and as such supports the IOPCSyncIO and IOPCAsyncIO2 interfaces. If the Read() or Write() methods on these interfaces are called for an itemID which has disappeared due to a batch being removed from the batch list, then the server should return the (already defined in Data Access) return status OPC_E_UNKNOWNITEMID.

An OPC batch server also supports the IOPCDataCallback interface. If an Advise is active on an itemID which disappears due to a batch being removed from the batch list a client receives OPC_E_UNKNOWNITEMID from the method IOPCDataCallback::OnDataChange()on this interface.

It is also possible that between the time an item was added to the group and it was set active the item has been removed from the server address space. The server shall return OPC_E_UNKNOWNITEMID for the initial update.

3.2.6 Use of Delimiter

In order to construct a fully qualified item ID, a client application must know the delimiter the OPC batch server uses to indicate the various objects in the hierarchy. Rather than define a specific delimiter (which may not be correct for all vendors and/or end users), this specification allows the vendor to specify the delimiter and provide that information to the client application. This specification assumes that only one delimiter is used by an OPC batch server.

As an example, a client could construct the list of parameters that are associated with a given RPE once the delimiter and the number of parameters are known. Given the RPE represented by the Item ID "OPCBBatchModel.B2000-A42.React", the knowledge that there are 4 parameters, and that the delimiter used by the server is ".", the client can construct the following 4 valid Item IDs:

OPCBBatchModel.B2000-A42.React.OPCBParameters.OPCBP1

OPCBBatchModel.B2000-A42.React.OPCBParameters.OPCBP2

OPCBBatchModel.B2000-A42.React.OPCBParameters.OPCBP3

OPCBBatchModel.B2000-A42.React.OPCBParameters.OPCBP4

A different server implementing the same namespace, but using a “\” as the delimiter, would allow for the following fully qualified item IDs:

OPCBBatchModel\B2000-A42\React\OPCBParameters\OPCBP1

OPCBBatchModel\B2000-A42\React\OPCBParameters\OPCBP2

OPCBBatchModel\B2000-A42\React\OPCBParameters\OPCBP3

OPCBBatchModel\B2000-A42\React\OPCBParameters\OPCBP4

3.3 OPC Batch Properties

The OPC batch namespace is further described using item properties at all levels of the namespace. The item properties are categorized as required or optional. Vendors may add vendor specific properties in addition to the required and optional properties.

The item properties are accessed using the IOPCItemProperties interface defined in the OPC Data Access Custom Interface Version 2.0 specification. In addition, property names have been defined for simplifying access to required and optional well-known properties. This addition is required for OPC batch servers.

3.3.1 Typical Use

Typical client use of property IDs would be to use the IOPCItemProperties::QueryAvailableProperties() method on the Item ID to determine what required, optional and vendor specific properties the OPC batch server supports for that item.

The required properties shall be supported for all items. Optional and vendor specific properties may be supported on an individual item basis and are discovered using IOPCItemProperties::QueryAvailableProperties.

3.3.2 How 'Property IDs' relate to ItemIDs

To obtain the data associated with a supported property, the property name is simply appended to the Item ID of the physical or batch model entity to create a new Item ID. For example, if Unit1 is the desired equipment and Name is the desired property, the associated new Item ID would be Unit1.Name.

3.3.3 Property List

The table below defines the OPC batch properties. Properties are identified using server assigned DWORD ID codes based on the OPC Data Access Custom Interface Specification 2.0 section 4.4.6. In addition the OPC batch specification defines property names to provide an alternate means of identification. The OPC batch specification reserves properties 400-999. In a future revision of the OPC Data Access Specification the ID Set 2 – Recommended Properties table will need to be revised to reflect this.

The Physical, Batch and RPE columns represent property sets. The scope of the physical property set is the physical model. The batch model uses two property sets, the batch property set and the RPE property set. The scope of the batch property set is the top level in the batch model (i.e. Batch ID). The scope of the RPE property set is all lower levels in the batch model, down to the OPCBParameters and OPCBResults collection items. This is done since the top level has extra properties required that would not be used at lower levels, therefore using two property sets reduces unused properties at a number of levels. The properties for the OPCBParameters and OPCBResults collection items are a subset of all the properties in Table 9- OPC Batch Properties. The set of properties a server supports for the OPCBParameters and OPCBResults collection items must be the same as the server supports for parameter and result items. Therefore a client may get the properties supported for one of the collection items and use that properties list for all the parameters and results in the model.

Recipe Procedural Elements refer to items in a procedure such as procedures, unit procedures, operations, phases and user defined model levels.

The property set columns (Physical, Batch and RPE) are used to specify which properties are present in the different models and at the different levels in the batch model. An "R" indicates the property is required. An "O" indicates the property is optional. An "-" indicates the property is not used for a property set. Server vendors should avoid using OPC optional property names and IDs for their vendor specific properties. Inside the scope of a property set all items shall have the same required properties.

Some batch properties have been derived from existing OPC Data Access and Alarms and Events properties. They have been assigned new property IDs since they have been given unique property names.

Notes:

1. When an Item ID is returned as a property value the server shall return a fully qualified item ID.
2. Property names are not case sensitive.

Table 9- OPC Batch Properties

ID	Property Name	Data Type of returned VARIANT	Standard Description	Physical	Batch	RPE
400	ID	VT_BSTR	A string identifying the name of the associated item. This could be an equipment ID, batch ID, internal ID or alias used by the batch system. This string is used to build the qualified item ID.	R	R	R
401	Value	<varies>	Derived from ID 2, "Item Value", in OPC Data Access Specification 2.0.	R	O	R
402	AccessRights	VT_I4	Derived from ID 5, "Item Access Rights", in OPC Data Access Specification 2.0	R	R	R
403	EU	VT_BSTR	Derived from ID 100, "EU Units", in OPC Data Access Specification 2.0	O	R	O
404	Description	VT_BSTR	Derived from ID 101, "Item Description", in OPC Data Access Specification 2.0	O	R	O
405	HighValueLimit	<varies>	Intended to be the highest value this item may take on.	O	O	O
406	LowValueLimit	<varies>	Intended to be the lowest value this item may take on.	O	O	O
407	TimeZone	VT_I4	Derived from ID 108, "Item Timezone", in OPC Data Access Specification 2.0	O	O	O
408	ConditionStatus	VT_BSTR	Derived from ID 300, "Condition Status", OPC Alarm & Event Specification 1.0	O	O	O
409	OPCBPhysicalModelLevel	VT_I4	The physical model level that is associated with this equipment. This value is an enumeration defined by OPCB_ENUM_PHYS .	R	--	--
410	OPCBBatchModelLevel	VT_I4	The IEC 61512-1 procedural model level that is associated with this batch or recipe procedural element. This value is an enumeration defined by OPCB_ENUM_PROC. Enumeration value 8 (OPCB_PROC_PROCEDURE) is used for the batch model level.	--	R	R

ID	Property Name	Data Type of returned VARIANT	Standard Description	Physical	Batch	RPE
411	RelatedBatchIDs	VT_ARRAY | VT_BSTR	A string identifying a batch or list of batches. The physical model levels of Site, Area, and Process Cells are intended to return a list of BatchIDs related to their ModelLevels. Generally Unit, Equipment Module, and Control Modules return a single BatchID related to the batch using the ModelLevel, however when they are shared use resources they may return a list.	R	--	--
412	Version	VT_BSTR	A string representing a version identifier for the associated item. The version format is dependent upon the server.	O	O	O
413	EquipmentClass	VT_BSTR	A string representing the class of the associated equipment. e.g., Reactor, Mixer, etc.	O	--	--
414	Location	VT_BSTR	A string representing a building or physical location where the item exists.	O	--	--
415	MaximumUserCount	VT_I4	A value representing the maximum number of concurrent users of the associated item. A value of –1 represents unlimited users.	O	--	--
416	CurrentUserCount	VT_I4	A value representing the current number of users of the associated item.	O	--	--
417	CurrentUserList	VT_ARRAY | VT_BSTR	A list of the Item IDs that are using the associated item.	O	--	--
418	AllocatedEquipmentList	VT_ARRAY | VT_BSTR	A list of equipment Item IDs that this item has allocated. Only the item performing the allocation lists the equipment item ID (i.e. the same allocation should not be reported in multiple lists).	O	O	O
419	RequesterList	VT_ARRAY | VT_BSTR	A list of Item IDs that are queued to allocate the associated item. Order implies precedence – first in the list is next to allocate.	O	O	O
420	RequestedList	VT_ARRAY | VT_BSTR	A list of Item IDs that this item has a pending allocation for.	O	O	O

ID	Property Name	Data Type of returned VARIANT	Standard Description	Physical	Batch	RPE
421	SharedByList	VT_ARRAY | VT_BSTR	A list of Item IDs that can share this item.	O	O	O
422	EquipmentState	VT_BSTR	A string representing the current state of the equipment.	O	--	--
423	EquipmentMode	VT_BSTR	A string representing the current mode of the equipment.	O	--	--
424	UpstreamEquipmentList	VT_ARRAY | VT_BSTR	A list of Item IDs representing the equipment from which material is directly received.	O	--	--
425	DownstreamEquipmentList	VT_ARRAY | VT_BSTR	A list of Item IDs representing the equipment to which material is directly sent.	O	--	--
426	EquipmentProceduralElementList	VT_ARRAY | VT_BSTR	A list of equipment procedural elements that this equipment can perform (e.g., for a unit – what equipment phases it can run). Since this specification has not defined equipment procedural elements this list cannot be assumed to be fully qualified Item IDs.	O	--	--
427	CurrentProcedureList	VT_ARRAY | VT_BSTR	A list of item IDs of the lowest level recipe procedural elements active on this equipment item.	O	--	--
428	TrainList	VT_ARRAY | VT_ARRAY | VT_BSTR	Defines the processing trains that exist in the equipment. Intended to only be populated for the Process Cell model level and no value returned for the other levels. Returned value is a two-dimensional array containing train names and the equipment item IDs for each train.	O	--	--
429	DeviceDataSource	VT_BSTR	A vendor specific string useful for addressing further information about this item. This may be used in conjunction with the DeviceDataServer property. For example this may be an item ID for an OPC data server.	O	--	--
430	DeviceDataServer	VT_BSTR	A vendor specific string useful for addressing further information about this item. This may be used in conjunction with the DeviceDataSource property. For example this may be an OPC data server name.	O	--	--
431	CampaignID	VT_BSTR	Production group of which this batch is a member.	--	O	--
432	LotIDList	VT_ARRAY | VT_BSTR	List of strings identifying the lots, which are related to this item.	O	O	O
433	ControlRecipeID	VT_BSTR	Control recipe that was used for this batch	--	O	--

ID	Property Name	Data Type of returned VARIANT	Standard Description	Physical	Batch	RPE
434	ControlRecipeVersion	VT_BSTR	Version of the control recipe that was used	--	O	--
435	MasterRecipeID	VT_BSTR	Master recipe that was used for this batch.	--	R	--
436	MasterRecipeVersion	VT_BSTR	Version of the master recipe that was used.	--	O	--
437	ProductID	VT_BSTR	Product which the execution of this batch will produce.	--	O	--
438	Grade	VT_BSTR	Grade of material being produced.	--	O	--
439	BatchSize	<varies>	Amount of material this batch will produce	--	R	--
440	Priority	VT_I4	Relative processing priority of the batch. Low numbers have the highest priority (e.g. priority 1 has a higher priority than priority 32).	--	O	--
441	ExecutionState	VT_BSTR	Current execution state using the vendor's state names.	--	R	R
442	IEC61512-1State	VT_I4	Current execution state using the example state names in IEC 61512-1. This permits vendor state names to be coerced into the example state names. This value is an enumeration defined by OPCB_ENUM_STATE.	--	O	O
443	ExecutionMode	VT_BSTR	Current execution mode using the vendor's mode names.	--	R	R
444	IEC61512-1Mode	VT_I4	Current execution mode using the example mode names in IEC 61512-1. This permits vendor mode names to be coerced into the example mode names. This value is an enumeration defined by OPCB_ENUM_MODE.	--	O	O
445	ScheduledStartTime	VT_DATE	Time when the batch, or other item, is scheduled to start.	--	O	O
446	ActualStartTime	VT_DATE	Time when the batch, or other item, actually started.	--	R	O
447	EstimatedEndTime	VT_DATE	Time when the batch, or other item, is planned to end.	--	O	O
448	ActualEndTime	VT_DATE	Time when the batch, or other item, actually ended.	--	R	O
449	OPCBPhysicalModelReference	VT_BSTR	The lowest level item ID in the OPCBPhysicalModel that encompasses all the equipment for this batch (this will usually be the process cell the batch is run in).	--	R	--

ID	Property Name	Data Type of returned VARIANT	Standard Description	Physical	Batch	RPE
450	EquipmentProceduralElement	VT_BSTR	The equipment procedural element that this item corresponds to (e.g., for a recipe phase what equipment phase does it correspond to). Since this specification has not defined equipment procedural elements this cannot be assumed to fully qualified Item ID.	--	O	O
451	ParameterCount	VT_I4	The number of parameters associated with this item.	--	O	R
452	ParameterType	VT_I4	The IEC 61512-1formula type: process input, process parameter or process output. This value is an enumeration defined by OPCB_ENUM_PARAM. Is only intended for parameter items.	--	O[1]	O[1]
453	ValidValues	VT_ARRAY | VT_BSTR	A list of the valid values for an item. Intended for cases where there is no contiguous range or for sets of strings. For example this could be used to return the list of material strings that can be entered into a field, or a list of non-contiguous integers that can be entered into a client data field.	O	O	O
454	ScalingRule	VT_BSTR	String containing any special scaling rules for this item. For example parameters may be scaled or not when a batch is scaled.	--	O	O
455	ExpressionRule	VT_BOOL	When the “Value” is a string this field is used to determine if the string is a literal or an expression. Primarily intended for parameters. 0 = literal 1 = expression	O	O[1]	O[1]
456	ResultCount	VT_I4	The number of results associated with this item.	--	O	R
457	EnumerationSetID	VT_I4	The vendor specific enumeration set ID associated with this item's "value" property. If the enumeration set ID does not exist the value is not an enumeration.	O	O	O

[1] - Intended for parameter items.

3.4 Enumeration Concept

OPC batch clients desire the ability to obtain state and other kinds of information from a batch server that will typically be implemented using enumerations. While it is possible for the OPC batch server to return this information in textual form, it does not provide an OPC batch client with the ability to perform custom actions based upon the contents of the string (locality / descriptions / naming differences, etc.). Returning textual information is also not as efficient as returning an enumeration value.

Extending this concept further, an OPC batch server may wish to utilize enumeration values for a number of the data items in its' namespace. An example would be the parameter list, where a particular parameter might correspond to an ingredient to add. Passing this information between client and server in textual form is both inefficient (data size of transmission) and error prone (spelling errors). Enumerations used in this context need to be defined by the OPC batch server.

Enumeration information consists of an Enumeration Set, an Enumeration Value, and an Enumeration String. The set and value fields are associated with the data values passed between client and server. The string value is the textual description of the enumeration that the client may query for display. This should be a localized string based upon the locality requested by the client.

Enumerations are extensible. Vendors may add enumerations to existing sets as well as add new enumeration sets.

Typical Use

The typical use of this information will be to translate from an enumeration set and value to a string that associates with the specified value. This translation may occur as needed during run-time as the client receives enumeration values or may occur at startup if the client wishes to obtain all the enumeration information from the server.

Examples

These are just examples and are not intended to impose any particular structure on any client or server implementation.

A client requests information about a particular item that the server supports. In addition to value information for this item, it is discovered that the value represents an enumeration and that it belongs to a particular enumeration set. The client takes both the enumeration set and value and passes them to the IOPCEnumerationSets::QueryEnumeration() method to obtain the textual representation for the enumeration. This is then displayed on an operator screen for the user.

Another client may desire that this run-time translation is not desirable, and will query the OPC batch server at connection about the set of enumerations that it supports and the possible values for each set. In this fashion, the client will maintain a lookup table of enumerations that may be encountered during operation.

Enumeration Sets

OPC batch reserves enumeration sets 0 through 99 and enumeration values 0-99 in each of the defined Enumeration Sets. The enumeration sets and enumeration values are extensible to allow for custom enumerations and custom enumeration sets. Custom enumeration sets may be assigned values greater than 99. Additional enumeration may be added to the OPC batch defined enumeration sets. As with enumeration sets, additional values for the OPC batch defined enumerations begin with values greater than 99. The only exceptions to enumeration set extension are the two enumeration sets corresponding to IEC61512-1Mode (OPCB_ENUM_MODE) and IEC 61512-1State (OPCB_ENUM_STATE). These enumeration sets are intended to provide clients with a well-defined state and mode value that conditional processing may take advantage of. Extending these enumeration sets will likely break clients who rely on these enumerations. When the IEC61512-1State and IEC61512-1Mode properties are used a valid enumeration should always be returned.

There is no implied meaning attached to the ordering of the enumeration values.

Table 10 - Enumerations

Enumeration Set	Enumeration Set Value	Enumeration	Enumeration Value
OPCB_ENUM_PHYS	0	OPCB_PHYS_ENTERPRISE	0
		OPCB_PHYS_SITE	1
		OPCB_PHYS_AREA	2
		OPCB_PHYS_PROCESSCELL	3
		OPCB_PHYS_UNIT	4
		OPCB_PHYS_EQUIPMENTMODULE	5
		OPCB_PHYS_CONTROLMODULE	6
		OPCB_PHYS_EPE	7
OPCB_ENUM_PROC	1	OPCB_PROC_PROCEDURE	0
		OPCB_PROC_UNITPROCEDURE	1
		OPCB_PROC_OPERATION	2
		OPCB_PROC_PHASE	3
		OPCB_PARAMETER_COLLECTION	4
		OPCB_PARAMETER	5
		OPCB_RESULT_COLLECTION	6
		OPCB_RESULT	7
		OPCB_BATCH	8
		OPCB_CAMPAIGN	9
OPCB_ENUM_STATE	2	OPCB_STATE_IDLE	0
		OPCB_STATE_RUNNING	1
		OPCB_STATE_COMPLETE	2
		OPCB_STATE_PAUSING	3
		OPCB_STATE_PAUSED	4
		OPCB_STATE_HOLDING	5
		OPCB_STATE_HELD	6
		OPCB_STATE_RESTARTING	7

Enumeration Set	Enumeration Set Value	Enumeration	Enumeration Value
		OPCB_STATE_STOPPING	8
		OPCB_STATE_STOPPED	9
		OPCB_STATE_ABORTING	10
		OPCB_STATE_ABORTED	11
		OPCB_STATE_UNKNOWN	12
OPCB_ENUM_MODE	3	OPCB_MODE_AUTOMATIC	0
		OPCB_MODE_SEMIAUTOMATIC	1
		OPCB_MODE_MANUAL	2
		OPCB_MODE_UNKNOWN	3
OPCB_ENUM_PARAM	4	OPCB_PARAM_PROCESSINPUT	0
		OPCB_PARAM_PROCESSPARAMETER	1
		OPCB_PARAM_PROCESSOUTPUT	2
Reserved for OPC Batch	5-99		
Vendor Specific	100+		

3.5 Compliance

A fully compliant OPC batch server has the following characteristics:

- ?? An OPC Data Access Server version 2.0 implementing all required interfaces,
- ?? Supports the IOPCBrowseServerAddressSpace optional custom OPC Data Access interface,
- ?? Supports the IOPCBatchServer , IEnumOPCBatchSummary, and IOPCEnumerationSets OPC batch custom interfaces,
- ?? Supports the defined Batch namespace and all the well-known ItemIDs, and
- ?? Supports the required properties in both the OPCBPhysicalModel and OPCBBatchModel, and adheres to the identified conventions for identifying and accessing properties.

3.6 OPC Data Access

The OPC Data Access specifications identify methods for reading and writing data. As the OPC batch server is additionally an OPC Data Access server, these methods and interfaces must be supported as identified in the OPC Data Access specifications.

The primary objective of the initial release of the OPC batch specification is the ability to share information between clients and server from different vendors. To this end, the entire OPC batch namespace must be readable. The ability to write value information to items in the OPC batch namespace is not a requirement, although vendors are free to provide write support for some or all items in their namespace.

Items in the OPC batch namespace that are not write-able will return OPC_E_BADRIGHTS which identifies the item as not write-able.

Additionally, there will be items in the OPC batch namespace that will not contain data. An example of which is the StartTime for a batch that has been added to the batch list, but has not yet started. For data items such as these, the concept of uninitialized data or data that is not meaningful may be supported. OPC batch servers that support the concept of non-meaningful data will return an error code of OPCB_E_NOT_MEANINGFUL to indicate [to the client application] that the data does not exist yet. For OPC batch servers that do not support the concept of non-meaningful data (i.e., there is an acceptable default value that is meaningful), the acceptable value may simply be returned.

Note that the IDataObject interface used by IOPCAsyncIO does not provide the ability to return error code information in the OnDataChange callback.

3.7 Reserved Name Summary

Table 11 - Well-known Item ID Summary

Reserved Names	Description
OPCBPhysicalModel	Hierarchy corresponding to the IEC 61512-1Physical Model
OPCBBatchModel	Hierarchy of batches and their recipe procedural elements (RPE)
OPCBBatchIDList	Convenience item returning a list of batch IDs
OPCBParameters	Collection point for parameters associated with a given recipe procedural element.
OPCBResults	Collection point for results associated with a given Recipe Procedural Element.

OPCBPn	Shorthand for anonymous parameter access
OPCBRn	Shorthand for anonymous result access

OPCBPhysicalModel, OPCBBatchModel, and OPCBBatchIDList always exist immediately under the root. OPCBParameters and OPCBResults are optional qualifiers for item IDs.

4 OPC Batch Server Custom Interface Quick Reference

This section includes a quick reference for the methods in the Custom Interface. These interfaces, their parameters, and behavior are defined in detail in section 5.

OPCBatchServer

- **IOPCBatchServer**
- **IEnumOPCBatchSummary**
- **IOPCEnumerationSets**

The Data Access Custom Interface Specification 2.0 defines these interfaces that are required by the OPC batch server specification:

OPCServer

- **IOPCServer**
- **IOPCServerPublicGroups (optional)**
- **IOPCBrowseServerAddressSpace (optional)**
- **IOPCItemProperties**
- **IConnectionPointContainer**
- **IOPCCommon**
- **IPersistFile (optional)**

OPCGroup

- **IOPCGroupStateMgt**
- **IOPCPublicGroupStateMgt (optional)**
- **IOPCASyncIO2**
- **IOPCItemMgt**
- **IConnectionPointContainer**
- **IOPCSyncIO**

EnumOPCItemAttributes

- **IEnumOPCItemAttributes**

4.1 OPC Batch Server Object

IOPCBatchServer

HRESULT	GetDelimiter(pszDelimiter)
HRESULT	CreateEnumerator(riid, ppUnk)

IEnumOPCBatchSummary

HRESULT	Next(celt, ppSummaryArray, pceltFetched)
HRESULT	Skip(celt)
HRESULT	Reset(void)
HRESULT	Clone(ppEnumBatchSummary)
HRESULT	Count(pcelt)

IOPCEnumerationSets

HRESULT	QueryEnumerationSets(pdwCount, ppdwEnumSetId, ppszEnumSetName)
HRESULT	QueryEnumeration(dwEnumSetId, dwEnumValue, pszEnumString)
HRESULT	QueryEnumerationList(dwEnumSetId, pdwCount, ppdwEnumValue, ppszEnumString)

5 OPC Batch Server Custom Interfaces

5.1 Overview

The OPC Data Access 2.0 specification sections 4.1, 4.2 and 4.3 cover the following topics:

4.1 - Overview of the OPC Custom Interface

4.2 – General Information

4.3 – Data Acquisition and Active State Behavior.

The OPC batch server has no additional requirements beyond these sections so this information is not repeated in this specification. It is assumed that the reader is familiar with these sections in the Data Access Specification.

5.2 OPCBatchServer Object

5.2.1 Overview

The OPCBatchServer object is the primary object that an OPC batch server exposes. The interfaces that this object provides include:

- ?? IUnknown
- ?? IOPCCommon
- ?? IOPCBatchServer
- ?? IOPCEnumerationSets
- ?? IEnumOPCBatchSummary

5.2.2 IUnknown

The server must provide a standard IUnknown Interface. Since this is a well-defined interface it is not discussed in detail. See the OLE Programmer's reference for additional information. This interface must be provided, and all functions implemented as required by Microsoft.

5.2.3 IOPCCommon

Other OPC servers such as data access share this interface design. It provides the ability to set and query a LocaleID that would be in effect for the particular client/server session. That is, the actions of one client do not affect any other clients.

As with other interfaces such as IUnknown, the instance of this interface for each server is unique. That is, an OPC Data Access server object and OPC batch server object might both provide an implementation of IOPCCommon. A client that is maintaining connections to both servers would, as with any other interface, use the interfaces on these two objects independently.

Since OPC batch servers must also be an OPC data access server and the OPC data access specification provides a detailed description of the IOPCCommon interface the definition is not repeated here. OPC batch servers must use the IOPCCommon interface defined in the OPC Data Server Specification version 2.0.

5.2.4 IOPCBatchServer

This is the main interface to the batch data of an OPC batch server. The OPC server is registered with the operating system as specified in the installation and registration chapter of this specification.

5.2.4.1 IOPCBatchServer::GetDelimiter

```
HRESULT GetDelimiter(
        [out, string]           LPWSTR  * pszDelimiter
        );
```

Description

Returns current status information for the server.

Parameters	Description
pszDelimiter	String containing the delimiter the server uses when returning fully qualified item IDs. An OPC batch server must use the same delimiter throughout the batch namespace.

Return Codes

Return Code	Description
E_FAIL	The operation failed.
E_OUTOFMEMORY	Not enough memory
E_INVALIDARG	An argument to the function was invalid.
S_OK	The operation succeeded.

Comments

Client must free the returned string.

5.2.4.2 IOPCBatchServer::CreateEnumerator

```
HRESULT CreateEnumerator(
        [in] REFIID riid,
        [out, iid_is(riid)] LPUNKNOWN* ppUnk
        );
```

Description

Create an enumerator for the batch server objects.

Parameters	Description
riid	The interface requested. Supported values are: IID_IEnumOPCBatchSummary.

ppUnk	Where to return the interface pointer. NULL is returned for any HRESULT other than S_OK

HRESULT Return Codes

Return Code	Description
S_OK	The function was successful.
S_FALSE	There is nothing to enumerate (e.g. there are no batches in the batch list).
E_OUTOFMEMORY	Not enough memory
E_INVALIDARG	An argument to the function was invalid (e.g. a bad riid parameter was passed.)
E_FAIL	The function was unsuccessful.

Comments

The client must release the returned interface pointer when it is done with it.

5.2.5 IEnumOPCBatchSummary

The IEnumOPCBatchSummary interface provides means to easily and efficiently access the summary batch list data.

This interface is returned only by IOPCBatchServer::CreateEnumerator() method. It is not available through QueryInterface.

The behavior of this interface is based upon the *IEnum** definition as defined in the Component Object Model. By convention *IEnum** interfaces contain the same member functions, but vary regarding the argument type that is being enumerated. However, in order to facilitate the Automation interface that will wrap this custom interface, an additional method Count() shall be added that will return the number of the batch presently in the batch list.

The batch summary is a snapshot.

Since enumeration is a standard interface this is described only briefly. See the Microsoft documentation concerning enumerators for a list and discussion of error codes.

5.2.5.1 IEnumOPCBatchSummary::Next

```
HRESULT Next(
        [in] ULONG celt,
        [out, size_is(,*pceltFetched)] OPCBATCHSUMMARY ** ppSummaryArray,
        [out] ULONG * pceltFetched
        );
```

Description

Fetch the next 'celt' items from the group.

Parameters	Description
celt	Number of items to be fetched.
ppSummaryArray	Array of OPCBATCHSUMMARY structures returned by the server.
pceltFetched	Number of items actually returned.

Comments

The client must free the returned OPCBATCHSUMMARY structures including the contained items.

5.2.5.2 IEnumOPCBatchSummary::Skip

HRESULT Skip(
[in] ULONG celt
);

Description

Skip over the next 'celt' attributes.

Parameters	Description
celt	Number of items to skip

Comments

5.2.5.3 IEnumOPCBatchSummary::Reset

HRESULT Reset(
void
);

Description

Reset the enumerator back to the first item.

Parameters	Description
Void	

Comments

5.2.5.4 IEnumOPCBatchSummary::Clone

HRESULT Clone(
[out] IEnumOPCBatchSummary** ppEnumBatchSummary
);

Description

Create a 2nd copy of the enumerator. The new enumerator will initially be in the same 'state' as the current enumerator.

Parameters	Description
ppEnumBatchSummary	Place to return the new interface

Comments

The client must release the returned interface pointer when it is done with it.

5.2.5.5 IEnumOPCBatchSummary::Count

HRESULT Count(
[out] ULONG *pcelt
);

Description

Return the number of items in the enumeration.

Parameters	Description
pcelt	Number of batches in the batch list.

5.2.6 IOPCEnumerationSets

5.2.6.1 IOPCEnumerationSets::QueryEnumerationSets

```
HRESULT QueryEnumerationSets(
        [out]DWORD *pdwCount,
        [out, size_is(,pdwCount)] DWORD **ppdwEnumSetId
        [out, string, size_is(,*pdwCount)] LPWSTR **ppszEnumSetName
         );
```

Description

Returns information about the enumeration sets that the Batch Server supports.

Parameters	Description
pdwCount	Count of Enumeration Sets in the server address space.
ppdwEnumSetId	Array of Enumeration Set IDs.
ppszEnumSetName	Array of Enumeration Set names corresponding to the Enumeration Set IDs.

Return Codes

Return Code	Description
E_FAIL	The operation failed.
E_OUTOFMEMORY	Not enough memory
S_OK	The operation succeeded.

Comments

The server should always include the standard enumeration sets in the returned values.

Clients must free the returned arrays.

5.2.6.2 IOPCEnumerationSets::QueryEnumeration

```
HRESULT QueryEnumeration(
        [in] DWORD dwEnumSetId,
        [in] DWORD dwEnumValue,
        [out, string] LPWSTR * pszEnumName
        );
```

Description

Returns information about custom enumerations

Parameters	**Description**
dwEnumSetId	Enumeration set to query
dwEnumValue	Enumeration value to query
pszEnumString	The returned string corresponding to the enumeration set and value.

Return Codes

Return Code	**Description**
E_FAIL	The operation failed.
E_OUTOFMEMORY	Not enough memory
E_INVALIDARG	An argument to the function was invalid.
S_OK	The operation succeeded.

Comments

Clients must free the returned string.

If either the enumeration set or value is out of range, E_INVALIDARG is returned.

5.2.6.3 IOPCEnumerationsSets::QueryEnumerationList

```
HRESULT QueryEnumerationList(
        [in] DWORD dwEnumSetId,
        [out] DWORD *pdwCount,
        [out, size_is(,*pdwCount)] DWORD       **ppdwEnumValue,
        [out, string, size_is(,*pdwCount)] LPWSTR ** ppszEnumString
        );
```

Description

Returns information about custom enumerations.

Parameters	Description
dwEnumSetId	Enumeration set to query
pdwCount	Count of the number of enumerations in the enumeration set
ppdwEnumValue	Array of Enumeration values associated with the specified enumeration set.
ppszEnumString	Array of Enumeration names corresponding to the enumeration set and values.

Return Codes

Return Code	Description
E_FAIL	The operation failed.
E_OUTOFMEMORY	Not enough memory
E_INVALIDARG	An argument to the function was invalid.
S_OK	The operation succeeded.

Comments

Clients must free the returned string.

If the enumeration set is out of range, E_INVALIDARG is returned.

6 Description of Data Types, Parameters and Structures

6.1 Structures and Masks

6.1.1 OPCBATCHSUMMARY

```
typedef struct {
        LPWSTR      szID;
        LPWSTR      szDescription;
        LPWSTR      szOPCItemID;
        LPWSTR      szMasterRecipeID;
        FLOAT       fBatchSize;
        LPWSTR      szEU;
        LPWSTR      szExecutionState;
        LPWSTR      szExecutionMode;
        FILETIME    ftActualStartTime
        FILETIME    ftActualEndTime
} OPCBATCHSUMMARY;
```

This structure used to communicate the status of each batch on the batch list to the client. This information is provided by the server in the IEnumOPCBatchSummary::Next() call.

Member	**Description**
szID	Vendor Identification of Batch
szDescription	Text string describing the batch
szOPCItemID	The OPC Item id for this batch object in the batch namespace
szMasterRecipeID	ID of Master Recipe
fBatchSize	Amount of material that this batch will produce
szEU	Engineering units for batch size
szExecutionState	Current execution state using vendor’s state names
szExecutionMode	Current execution mode using vendor's mode names
ftActualStartTime	Time when the batch actually started.
ftActualEndTime	Time when the batch actually ended.

7 Installation Issues

Since an OPC batch server is also an OPC data access server the installation is the same for both with the addition that OPC batch servers must register as both an OPC data access server and an OPC batch server with the component category managers. Refer to the OPC Data Access Specification version 2.0 section 5.

For OPC batch specification version 1.0 the

component category descriptor is: "OPC Batch Server Version 1.0"

and the

component category ID is: { a8080da0-e23e-11d2-afa7-00c04f539421}

It is expected that a server will first create any category it uses and then will register for that category. Unregistering a server should cause it to be removed from that category. For additional information see Microsoft documentation for ICatRegister.

8 Summary of OPC Error Codes

The OPC batch specification error codes beyond those defined in the OPC Data Access Specification version 2.0 are defined in Appendix B.

9 Appendix A – OPC Batch Custom IDL Specification

The current files require MIDL compiler 5.00 or later.

Use the command line MIDL /ms_ext /c_ext /app_config opcbc.idl.

The resulting opcbc.h file can be included in clients and servers. The resulting opcbc_i.c file defines the interface Ids and can be **Linked** into clients and servers that include opcbc.h.

Alternatively, clients and servers may choose to use the Type Library that is embedded in the resource of the proxy/stub DLL (opcbc_ps.dll). In Visual C++ this is accomplished with the #import statement:

```
#import “opcbc_ps.dll” exclude(“_FILETIME”)
using namespace OPC_BATCH
```

NOTE: This IDL file and the Proxy/Stub generated from it should NEVER be modified in any way. If you add vendor specific interfaces to your server (which is allowed) you must generate a SEPARATE vendor specific ProxyStub DLL to marshal only those interfaces.

```
// opcbc.idl
//
// REVISION: 01/17/2000
// VERSIONINFO  1.0.0.0
//

import "oaidl.idl";

typedef struct tagOPCBATCHSUMMARY {
  [string] LPWSTR   szID;
  [string] LPWSTR   szDescription;
  [string] LPWSTR   szOPCItemID;
  [string] LPWSTR   szMasterRecipeID;
           FLOAT    fBatchSize;
  [string] LPWSTR   szEU;
  [string] LPWSTR   szExecutionState;
  [string] LPWSTR   szExecutionMode;
           FILETIME ftActualStartTime;
           FILETIME ftActualEndTime;
} OPCBATCHSUMMARY;



// Define OPC Batch Interfaces
[
  uuid("8BB4ED50-B314-11d3-B3EA-00C04F8ECEAA"),
  helpstring("IOPCBatchServer Interface"),
  pointer_default(unique)
]
interface IOPCBatchServer : IUnknown
{
  HRESULT GetDelimiter (
    [out, string]           LPWSTR  * pszDelimiter
    );
```

```
  HRESULT CreateEnumerator(
    [in]        REFIID      riid,
    [out, iid_is(riid)] LPUNKNOWN * ppUnk
    );
};


// Define OPC Batch Summary Enumeration Interfaces
[
  uuid("a8080da2-e23e-11d2-afa7-00c04f539421"),
  helpstring("IEnumOPCBatchSummary"),
  pointer_default(unique)
]
interface IEnumOPCBatchSummary : IUnknown
{
  HRESULT Next(
    [in]                ULONG   celt,
    [out, size_is(,*pceltFetched)]    OPCBATCHSUMMARY ** ppSummaryArray,
    [out]               ULONG * pceltFetched
    );

  HRESULT Skip(
    [in]                ULONG   celt
    );

  HRESULT Reset(
    void
    );

  HRESULT Clone(
    [out]               IEnumOPCBatchSummary ** ppEnumBatchSummary
    );

HRESULT Count(
    [out]               ULONG * pcelt
    );
};


// Define OPC Enumeration Set Interfaces
[
  uuid("a8080da3-e23e-11d2-afa7-00c04f539421"),
  helpstring("IOPCEnumerationSets Interface"),
  pointer_default(unique)
]
interface IOPCEnumerationSets : IUnknown
{
  HRESULT QueryEnumerationSets(
    [out]                               DWORD * pdwCount,
    [out, size_is(,*pdwCount)]          DWORD **  ppdwEnumSetId,
    [out, string, size_is(,*pdwCount)]  LPWSTR  **  ppszEnumSetName
    );

  HRESULT QueryEnumeration(
    [in]                                DWORD   dwEnumSetId,
    [in]                                DWORD   dwEnumValue,
```

```
    [out, string]                       LPWSTR  * pszEnumName
    );

  HRESULT QueryEnumerationList(
    [in]                                DWORD   dwEnumSetId,
    [out]                               DWORD * pdwCount,
    [out, size_is(,*pdwCount)]          DWORD **  ppdwEnumValue,
    [out, string, size_is(,*pdwCount)]  LPWSTR  **  ppszEnumName
    );
}


[
  uuid("a8080da4-e23e-11d2-afa7-00c04f539421"),
  version(1.0),
  helpstring("opc_batch 1.0 Type Library")
]
library OPC_BATCH
{
  importlib("stdole32.tlb");
  importlib("stdole2.tlb");

  interface IOPCBatchServer;
  interface IEnumOPCBatchSummary;
  interface IOPCEnumerationSets;
};
```

10 Appendix B OPCBatchError.h

```
/*++
Module Name:
 OpcBatchError.h
Author:
OPC Batch Committee

Revision History:
Release 1.0
    initial version for 1.0 spec

--*/

/*
Code Assignements:
  0000 to 0200 are reserved for Microsoft use
  (although some were inadverdantly used for OPC 1.0 errors).

  0200 to 8000 are reserved for future OPC use.
    of these, 0300 to 03FF are reserved for future OPC Batch use

  8000 to FFFF can be vendor specific.

*/


//
// MessageId: OPCB_E_NOT_MEANINGFUL
//
// MessageText:
//
//  The data is not meaningful at the present time
//
#define OPCB_E_NOT_MEANINGFUL              ((HRESULT)0xC0040300L)
```

11 Appendix C – OPCBatchDef.h

```
/*++

  Module Name:

 OPCBatchDef.h

  Abstract:

 Macros defined for OPC Batch Clients and Servers

  Author:

 OPC Batch Committee

  Revision History:

  --*/

#ifndef OPCBATCHDEF_H
#define OPCBATCHDEF_H


// Define the various Batch Enumeration Sets
//
//   Custom Enumeration Set IDs start at 100
//   Custom Enumeration Values for any of the defined Enumeration
//     sets may be appended.  These custom enumeration values start
//     at 100.
//
//   The enumeration values and corresponding localized string
//     representation are returned via the IOPCEnumerationSets
//     interface methods.


// OPC Batch Enumeration Sets
#define OPCB_ENUM_PHYS         0
#define OPCB_ENUM_PROC         1
#define OPCB_ENUM_STATE        2
#define OPCB_ENUM_MODE         3
#define OPCB_ENUM_PARAM        4


// OPC Batch Physical Model Level Enumeration
#define OPCB_PHYS_ENTERPRISE   0
#define OPCB_PHYS_SITE         1
#define OPCB_PHYS_AREA         2
#define OPCB_PHYS_PROCESSCELL     3
#define OPCB_PHYS_UNIT         4
#define OPCB_PHYS_EQUIPMENTMODULE 5
#define OPCB_PHYS_CONTROLMODULE   6
#define OPCB_PHYS_EPE        7
```

```
// OPC Batch Procedural Model Level Enumeration
#define OPCB_PROC_PROCEDURE       0
#define OPCB_PROC_UNITPROCEDURE   1
#define OPCB_PROC_OPERATION       2
#define OPCB_PROC_PHASE        3
#define OPCB_PROC_PARAMETER_COLLECTION 4
#define OPCB_PROC_PARAMETER 5
#define OPCB_PROC_RESULT_COLLECTION 6
#define OPCB_PROC_RESULT 7
#define OPCB_PROC_BATCH 8
#define OPCB_PROC_CAMPAIGN 9



// OPC Batch IEC 61512-1State Index Enumeration
#define OPCB_STATE_IDLE        0
#define OPCB_STATE_RUNNING     1
#define OPCB_STATE_COMPLETE       2
#define OPCB_STATE_PAUSING     3
#define OPCB_STATE_PAUSED      4
#define OPCB_STATE_HOLDING     5
#define OPCB_STATE_HELD        6
#define OPCB_STATE_RESTARTING     7
#define OPCB_STATE_STOPPING       8
#define OPCB_STATE_STOPPED     9
#define OPCB_STATE_ABORTING       10
#define OPCB_STATE_ABORTED     11
#define OPCB_STATE_UNKNOWN     12

// OPC Batch IEC 61512-1Mode Index Enumeration
#define OPCB_MODE_AUTOMATIC       0
#define OPCB_MODE_SEMIAUTOMATIC   1
#define OPCB_MODE_MANUAL       2
#define OPCB_MODE_UNKNOWN      3

// OPC Batch Parameter Type Enumeration
#define OPCB_PARAM_PROCESSINPUT   0
#define OPCB_PARAM_PROCESSPARAMETER 1
#define OPCB_PARAM_PROCESSOUTPUT  2

#endif
```

12 Appendix D - OPCBatchProps.h

```
/*++
Module Name:
 OPCBatchProps.h
Author:
 OPC Batch Committee

Revision History:
17-Jan-2000   Revision  1.0    Created

--*/

/*
 Property ID Code Assignments:
  400 to 999 are reserved for OPC Batch use
*/

#ifndef __OPCBATCHPROPS_H
#define __OPCBATCHPROPS_H

#define OPC_PROP_B_ID                           400
#define OPC_PROP_B_VALUE                        401
#define OPC_PROP_B_RIGHTS                       402
#define OPC_PROP_B_EU                           403
#define OPC_PROP_B_DESC                         404
#define OPC_PROP_B_HIGH_VALUE_LIMIT             405
#define OPC_PROP_B_LOW_VALUE_LIMIT              406
#define OPC_PROP_B_TIME_ZONE                    407
#define OPC_PROP_B_CONDITION_STATUS             408
#define OPC_PROP_B_PHYSICAL_MODEL_LEVEL         409
#define OPC_PROP_B_BATCH_MODEL_LEVEL            410
#define OPC_PROP_B_RELATED_BATCH_IDS            411
#define OPC_PROP_B_VERSION                      412
#define OPC_PROP_B_EQUIPMENT_CLASS              413
#define OPC_PROP_B_LOCATION                     414
#define OPC_PROP_B_MAXIMUM_USER_COUNT           415
#define OPC_PROP_B_CURRENT_USER_COUNT           416
#define OPC_PROP_B_CURRENT_USER_LIST            417
#define OPC_PROP_B_ALLOCATED_EQUIPMENT_LIST     418
#define OPC_PROP_B_REQUESTER_LIST               419
#define OPC_PROP_B_REQUESTED_LIST               420
#define OPC_PROP_B_SHARED_BY_LIST               421
#define OPC_PROP_B_EQUIPMENT_STATE              422
#define OPC_PROP_B_EQUIPMENT_MODE               423
#define OPC_PROP_B_UPSTREAM_EQUIPMENT_LIST      424
#define OPC_PROP_B_DOWNSTREAM_EQUIPMENT_LIST    425
#define OPC_PROP_B_EQUIPMENT_PROCEDURAL_ELEMENT_LIST   426
#define OPC_PROP_B_CURRENT_PROCEDURE_LIST       427
#define OPC_PROP_B_TRAIN_LIST                   428
#define OPC_PROP_B_DEVICE_DATA_SOURCE           429
#define OPC_PROP_B_DEVICE_DATA_SERVER           430
#define OPC_PROP_B_CAMPAIGN_ID                  431
#define OPC_PROP_B_LOT_ID_LIST                  432
```

```
#define OPC_PROP_B_CONTROL_RECIPE_ID            433
#define OPC_PROP_B_CONTROL_RECIPE_VERSION       434
#define OPC_PROP_B_MASTER_RECIPE_ID             435
#define OPC_PROP_B_MASTER_RECIPE_VERSION        436
#define OPC_PROP_B_PRODUCT_ID                   437
#define OPC_PROP_B_GRADE                        438
#define OPC_PROP_B_BATCH_SIZE                   439
#define OPC_PROP_B_PRIORITY                     440
#define OPC_PROP_B_EXECUTION_STATE              441
#define OPC_PROP_B_IEC61512_1_STATE             442
#define OPC_PROP_B_EXECUTION_MODE               443
#define OPC_PROP_B_IEC61512_1_MODE              444
#define OPC_PROP_B_SCHEDULED_START_TIME         445
#define OPC_PROP_B_ACTUAL_START_TIME            446
#define OPC_PROP_B_ESTIMATED_END_TIME           447
#define OPC_PROP_B_ACTUAL_END_TIME              448
#define OPC_PROP_B_PHYSICAL_MODEL_REFERENCE     449
#define OPC_PROP_B_EQUIPMENT_PROCEDURAL_ELEMENT 450
#define OPC_PROP_B_PARAMETER_COUNT              451
#define OPC_PROP_B_PARAMETER_TYPE               452
#define OPC_PROP_B_VALID_VALUES                 453
#define OPC_PROP_B_SCALING_RULE                 454
#define OPC_PROP_B_EXPRESSION_RULE              455
#define OPC_PROP_B_RESULT_COUNT                 456
#define OPC_PROP_B_ENUMERATION_SET_ID           457

#endif
```